uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática

2019-2020

Trabajo Fin de Grado

"Adaptación al terreno del proceso de marcha de un robot hexápodo"

David Gil López

Tutor

Agapito Ismael Ledezma Espino

Leganés, 25/9/2020

Resumen

A día de hoy las tareas de rescate son un tipo de acción bastante peligrosa ya que quien la realiza se expone a ser golpeado por desprendimientos y otros elementos que no se pueden controlar.

En este documento se estudiará la posible aplicación de un robot hexápodo para estas tareas a través de la optimización de su marcha en terreno irregular. Para esto se realizará un análisis de la situación actual de la robótica, tratando temas como su historia, sus clasificaciones, paradigmas y también se hará especial hincapié en la competición llamada RoboCup así como su impacto en el mundo de la robótica y la inteligencia artificial. Además de esto se hablará de simuladores.

Después se tratarán los métodos utilizados para optimizar el sistema de marcha del robot hexápodo sobre terreno irregular utilizando algoritmos genéticos evolutivos para optimizar los parámetros de una función seno a través de la cual se definirán los valores de actuación de los motores del robot. Esto se llevará a cabo en el simulador Webots.

Finalmente se tratará el modelo final el cual ha sido optimizado, por qué se ha optado por él y qué conclusiones se han sacado tanto de la investigación sobre el contexto y las soluciones del trabajo como del desarrollo del trabajo en sí.

Palabras clave: Algoritmos Genéticos, Optimización, Marcha, Robot hexápodo, Terreno irregular.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Capítulo 1: Introducción

1.1 Motivación

A día de hoy, el método de desplazamiento más común que se encuentra en la gran mayoría de vehículos es el basado en sistemas locomotores de tipo rodantes, pero esto no siempre es aplicable o no es suficientemente bueno para todos los entornos, por eso se están estudiando alternativas.

Uno de los estudios que a día de hoy plantean alternativas a los sistemas locomotores de tipo rodantes es el de los robots zoomórficos del tipo caminador. Estos están basados en patas, que se adaptan mucho mejor que las ruedas a las irregularidades del terreno.

Una de las aplicaciones que tienen los robots con métodos de desplazamiento alternativos son la investigación de entornos de riesgo, los cuales no son del todo accesibles o suponen un reto para otros robots como puedan ser terrenos accidentados, volcanes o incluso, si se aplican a rescate, edificios destruidos por diversas razones.

Este trabajo está orientado a encontrar un método a través del cual un robot hexápodo, (Mantis), se pueda desplazar de forma estable a través de terreno irregular. Esto se realizará en el simulador Webots utilizando el terreno irregular ofrecido por el propio programa como campo de pruebas y evaluación.

1.2 Objetivos

Para definir los objetivos de este trabajo, primero se definirá un objetivo general el cual se descompondrá en una serie de objetivos específicos. Por un lado, el objetivo general del trabajo es:

- **Optimizar el proceso de marcha de un robot hexápodo en terreno irregular.** Para ello se deberá definir y optimizar el comportamiento de los motores del robot hexapodo "Mantis" en el simulador Webots.

Tras definir esto, los objetivos específicos de este trabajo son:

- **Adquirir las competencias necesarias.** Aprender sobre el campo de los algoritmos genéticos y en el uso del simulador Webots para poder hacer frente a las necesidades que haya durante el desarrollo del trabajo.

- **Implementar el controlador.** Realizar una implementación del controlador óptimo, la cual tenga parámetros optimizables a través de un algoritmo genético así como el propio algoritmo.

- **Entrenar el algoritmo genético.** Realizar el entrenamiento y la experimentación necesaria para que encuentre una solucion que satifaga el objetivo princial

- **Evaluar los resultados.** Evaluar, a través de los datos del entrenamiento y del individuo resultado al acabar el entrenamiento, los resultados obtenidos y sacar conclusiones sobre ello.

1.3 Metodología

La metodología seguida a la hora del desarrollo de este proyecto es una basada en desarrollo ágil, ya que el desarrollo del trabajo ha ido cumpliendo objetivos cada vez más ambiciosos en función de la etapa en la que se encontrase el proyecto, empezando por un programa simple que fuera capaz de mover el robot hasta la implementación con algoritmos genéticos que ha resultado en el modelo con el cual se ha entrenado el individuo final.

Para ser exactos, se puede hablar de una metodología similar a SCRUM salvando ciertas distancias, ya que este trabajo ha sido desarrollado por un único individuo en vez de un equipo y tampoco se ha enseñado los avances a un cliente sino al tutor del propio trabajo. Aun así, se puede hablar de una cierta similitud ya que se han mantenido los *sprints*, la planificación de los mismos, su revisión y un análisis en retrospectiva para mejorar de cara a la siguiente iteración.

Los *sprints* han sido mayormente de 2 semanas, en las cuales se ha avanzado en el proyecto acorde a los objetivos planificados en la reunión anterior, de forma que se ha ido progresando sin parones innecesarios y gracias a estas mismas reuniones se ha podido cambiar sobre la marcha la forma en el que se ha planteado el proyecto varias veces, permitiendo así obtener un producto final más refinado y cercano al objetivo planteado inicialmente.

1.4 Marco legal

Referente al marco legal que cubre este trabajo se pueden hacer dos distinciones, la que hace referencia a la posible implementación del robot y trata la robótica en su conjunto y, por otro lado, la que cubre los temas asociados con el *software*, así como el uso de datos y demás temas relacionados.

En cuanto a la legislación referente a la robótica, España se acoge a la normativa que propone el Parlamento Europeo sobre robótica e inteligencia artificial. Esta convención recoge una serie de leyes que se resumen en tres conceptos [30]:

- Es necesario llevar un registro de los robots existentes.
- Es necesario tener un sistema que de forma objetiva evalúe la responsabilidad de los posibles daños ocasionados y un sistema de seguros.
- Debido al cambio en el mercado laboral es necesario tener un régimen que regule la cotización a seguridad social por los dueños de los robots.

En cuanto al propio *software* que comanda al robot, este se rige por la normativa que rige el resto de *software* de inteligencia artificial, es decir, el tratamiento de datos con el que se lleva a cabo el entrenamiento, las leyes que defienden a los usuarios del *software* final y a la propiedad intelectual del propio sistema. Referente a estos temas tenemos las siguiente normativas y leyes [30]:

- Ley de patentes y marcas
- Ley de propiedad intelectual
- Ley Orgánica de protección de datos
- Ley de defensa de consumidores y usuarios

1.5 Estructura

Este documento está compuesto por seis capítulos y un anexo estructurados de la siguiente forma:

- El Capítulo 1 consta de la introducción, donde se trata la motivación, los objetivos del trabajo, la metodología seguida, la información legal referente a la normativa que sigue la Unión Europea sobre la robótica, las leyes que regulan la protección de datos y temas relacionados que afectan a la inteligencia artificial que optimiza la marcha del robot y por último esta estructura.

- El Capítulo 2 es el estado del arte, donde se hace una introducción a la robótica, se explica la estructura que siguen los robots a nivel morfológico y en qué paradigmas se pueden clasificar. Además se habla de la RoboCup, haciendo especial hincapié en el apartado de rescate, donde se puede aplicar el resultado de este trabajo. Por último, se habla de simuladores explicando Webots en específico y hablando del robot Mantis.

- El Capítulo 3 es el análisis y el diseño del sistema empleado, donde se hablará del *software* utilizado y el razonamiento de este, así como del análisis de requisitos que se ha realizado para formalizar los objetivos del trabajo.

- El Capítulo 4 es donde se analizan los diferentes enfoques a través de los cuales se ha realizado un trabajo y se realizará una explicación de las decisiones de diseño que los han motivado y los resultados. Además de esto se añadirán diferentes datos que representen la evolución de los individuo a lo largo de las generaciones y un apartado en el que se explicará el modelo final por separado.

- El Capítulo 5 recoge la fase de planificación y el presupuesto del trabajo, contando el precio de los servicios utilizados, el ordenador desde el que se ha trabajado y el sueldo correspondiente a un ingeniero dadas las horas de trabajo.

- El Capítulo 6 cubre las conclusiones sacadas en la realización del trabajo y de los resultados obtenidos en el mismo.

- El anexo indicado previamente es el resumen en inglés del documento.

Capítulo 2: Estado del arte

2.1 Introducción a la robótica

A día de hoy la robótica es una técnica muy extendida y utilizada para agilizar trabajos que antiguamente requerían mucha fuerza humana, haciéndolos de forma más rápida, eficiente y barata. Además se utiliza para realizar tareas de asistencia en diferentes lugares como por ejemplo el robot *Sophia* en residencias de la tercera edad y para investigar lugares de riesgo como por ejemplo escenarios de desastres, lugares extremos como volcanes o incluso para la exploración de otros planetas. Este trabajo trata de cubrir este último apartado, ya que se plantea la adaptación del sistema de marcha de un robot hexápodo a terreno irregular.

A continuación, se procederá a realizar una revisión de los conceptos sobre la robótica, algunas de sus aplicaciones y el uso de simuladores en la misma.

2.1.1 Análisis etimológico y definición de robótica

La palabra robot proviene de la palabra checa *robota* que significa trabajo o labor. Esta palabra la creó Josef Čapek para que su hermano, el autor Karel Čapek la usara y la popularizara con su obra *"R.U.R. (Robots Universales Rossum)"*. Esta palabra pasó a sustituir la palabra autómata, que había sido empleada previamente por el mismo autor en el relato *"Opilec"*.

Por otro lado, la persona que empezó a usar propiamente la palabra robótica para definir el campo fue Isaac Asimov, un autor de ciencia ficción y divulgación científica, siendo robótica el campo que agrupa el estudio de la mecatrónica, la física y la matemática. Este mismo autor, dentro de sus relatos *"I, Robot*" [58], enuncia una serie de leyes que deben cumplir los robots, las 3 Leyes de la robótica:

1. “Un robot no puede hacer daño a un ser humano o, por inanición, permitir que un ser humano sufra daño. “
2. “Un robot debe obedecer las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la Primera Ley. “
3. “Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la Primera Ley o la Segunda Ley.”

Además de estas, en su novela *“Robots and Empire”* [57], añadió una cuarta ley que debía tener prioridad sobre las otras y que se conoció como la Ley 0: “Un robot no puede lastimar a la humanidad o, por falta de acción, permitir que la humanidad sufra daño.” Esta se consideró que tenía más prioridad que el resto porque daba prioridad al bien común antes que al de los individuos.

Si se busca una definición más formal, según la RAE [59], la robótica se define como *“Técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales.”*

2.1.2 Historia de la robótica

La robótica [2] ha tenido una gran evolución a lo largo de los años, desde escritos en textos chinos y pequeños instrumentos de madera a elementos más actuales como brazos robóticos o robots bípedos. A continuación, se procederá a explicar una pequeña cronología con los elementos más destacables de este campo a lo largo de toda la historia.

- Una de las primeras veces que se escribe sobre una figura humanoide data alrededor de 350 A.C. Esta se encuentra en el texto Lie Zi escrito por el taoísta Lie Yukou [1]. En esta se habla sobre Yan Shi un "artífice" que se encontró con el Rey Mu de Zhou.

- Por estos años se cree que Arquitas de Tarento, un inventor pitagórico, creó una paloma de madera que gracias a un sistema de contrapesos y a aire contenido en su interior era capaz de volar.

- Más adelante, sobre el siglo I A.C. existen descripciones de alrededor de 100 máquinas en los textos y libros de Herón de Alejandría [3], al que se le atribuye el primer libro de robótica de la historia. Entre las máquinas descritas por Herón, podemos encontrar herramientas de observación como la dioptra, la primera máquina de vapor (eolipila), que se puede ver en la figura 1, o la fuente de Herón.

Fig. 1: Una eolipila moderna [33]

- En 1206, el escritor árabe Al-Jazari[6] escribió *"El libro del conocimiento de dispositivos mecánicos ingeniosos"*, un libro en el que se describen 100 dispositivos mecánicos entre los que se describen más de 80 tipos de naves, con instrucciones de cómo construirlos.

- Más adelante, en 1495, el polímata florentino Leonardo Da Vinci [7] diseñó un robot humanoide llamado Caballero Mecánico [8], que se puede ver en la figura 2, el cual actualmente se ha recreado siguiendo sus diseños y se ha demostrado que es completamente funcional.

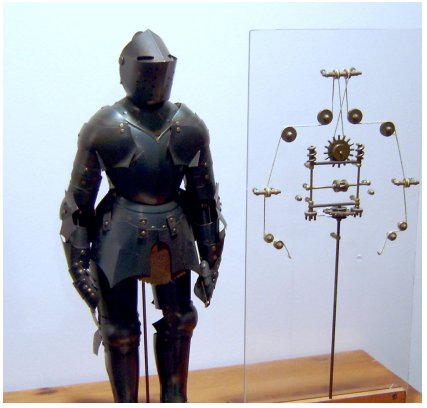


Fig. 2: Recreación del Caballero Mecánico [34]

- En 1738 el ingeniero e inventor Jacques de Vaucanson diseñó y creó un pato mecánico que podía comer, agitar sus alas y excretar. Este robot iba movido por varios mecanismos unidos a una rueda la cual al girar movía los diferentes mecanismos [4].

- Sobre el año 1800 el inventor e ingeniero japonés Tanaka Hisashige[9] diseñó una serie de juguetes llamados *karakuri dolls*, los cuales eran autónomos, movidos por muelles, circuitos neumáticos o circuitos hidráulicos, siendo capaces de hacer movimientos relativamente complejos.

- En 1921, como se había mencionado previamente, apareció el primer autómata llamado robot en el libro RUR.

- Entre los años 1937 y 1938 se diseñó Elektro [10], un robot construido por Westinghouse Electric Corporation que se puede ver en la figura 3. Este podía andar por comandos de voz, decir alrededor de 700 palabras, fumar, hinchar globos y mover sus brazos y cabeza.

Fig. 3: Elektro [35]

- En 1942 se publicó la historia de ciencia ficción donde se encuentran las leyes de la robótica mencionadas previamente.

- Elmer y Elsie [11] fueron dos tortugas robóticas construidas por el neurólogo William Grey Walter entre 1948 y 1949. Estos fueron los primeros robots en la historia que estaban programados para actuar simulando pensamientos de cerebros biológicos con intención de tener libre albedrío. Estas deambulaban y tenían sensores de luz y táctiles a los cuales respondían, desplazándose hacia luces y esquivando los obstáculos que se encontraran.

- En 1956 la compañía Unimation creó el primer robot comercial, Unimate [12] (figura 4) y más adelante, en 1961, esta misma compañía lo instaló, siendo el primer robot industrial.

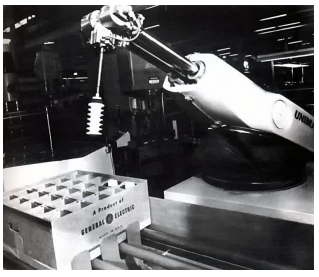


Fig. 4:Unimate 1961 [36]

- En 1963 se creó el primer robot de control de palés.

- En 1971 el Mars 3 aterrizó en Marte, siendo este el primer robot soviético que lo consiguió de forma exitosa, aunque se perdió la conexión con él poco después. 5 años después el Viking 1 de la nasa aterrizó en Marte.

- En 1973 se creó el Famulus, el primer robot con 6 ejes electromecánicos.

- En 1975 se creó PUMA, un brazo robótico programable para manipulación de elementos. Este brazo fue creado por Unimation.

- En 1982 Asimov amplió las 3 leyes de la robótica como se mencionó previamente con la Ley 0.

- En 1992 se fundó Boston Dynamics. [50]

- En el año 2000 se presentó por primera vez el robot ASIMO, el cual era capaz de desplazarse de forma bípeda e interactuar con personas. [50]

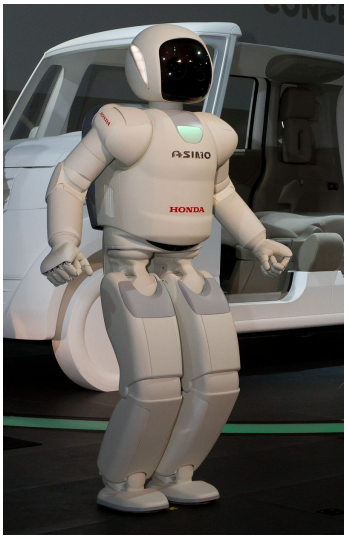


Fig. 5: Robot ASIMO [37]

- En el 2004 se lanzó el primer robot NAO[13] y en 2007 sustituyó al robot aibo como plataforma estándar en la RoboCup.

Fig. 6: Robot NAO [38]

- En 2015 se creó Sophia [14], un robot humanoide con capacidad para interactuar con personas, reconocer caras y simular expresiones. Fue diseñada como compañera en residencias de ancianos.

2.1.3 Clasificación de robots según su estructura

Para realizar una clasificación [2] primero se tiene que analizar su estructura, la cual no siempre es una tarea trivial, ya que existe el metamorfismo que ha aparecido recientemente. El metamorfismo permite que la estructura de los robots varíe para ser más flexible y adaptarse a más tareas. Esto puede suceder a un nivel elemental donde se varíen herramientas o a un nivel más complejo donde subsistemas enteros del robot se modifiquen.

Sabiendo esto para hacer una clasificación estructural de los robots se utiliza su arquitectura como base, dando lugar a las siguientes clasificaciones:

2.1.3.1 Poliarticulados

La arquitectura poliarticulada agrupa los robots que son estáticos y realizan acciones a través de herramientas en una determinada área de trabajo de una o más dimensiones con un grado de libertad limitado.

Un ejemplo de este tipo de arquitectura es el sistema quirúrgico Da Vinci (figura 7), una herramienta que permite a cirujanos operar a través de un sistema de control con mucha precisión, eliminando posibles temblores. Está compuesto por varios brazos robóticos poliarticulados que se estructuran sobre una cama de hospital permitiendo al cirujano que controla el robot realizar acciones muy precisas sobre el paciente.[60]

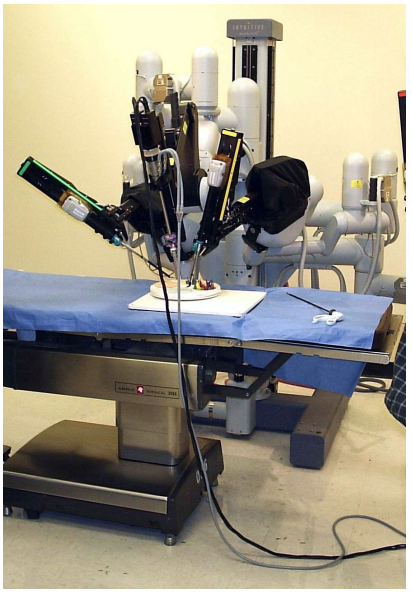


Fig. 7:Sistema quirúrgico DaVinci [39]

2.1.3.2 Móviles

Los robots que tienen esta arquitectura tienen una gran movilidad horizontal, su sistema locomotor está basado en ruedas y suelen decidir sus caminos mediante información externa recibida o a través de sus propios sensores. Estos robots son capaces de esquivar obstáculos y tienen una inteligencia relativamente alta.

Un ejemplo de este tipo de arquitectura son los roombas (figura 8), aspiradoras inteligentes que limpian superficies moviéndose por sí mismas trazando caminos en los cuales esquivan los obstáculos que suponen el mobiliario. [61]

Fig. 8: Roomba [40]

2.1.3.3 Androides

Esta arquitectura cubre los robots que están basados parcial o completamente en el ser humano. Es un campo que está relativamente poco avanzado y centrado mayormente en estudio y experimentación. El mayor problema que supone esta arquitectura es realizar las tareas que se manden a los robots mientras mantienen el equilibrio.

Un ejemplo de este tipo de arquitectura es el robot Atlas de Boston Dynamics (figura 9). Este robot no solo es capaz de estar erguido sobre dos patas sino que también es capaz de correr, hacer el pino, volteretas, saltos y, en palabras de los propios creadores, parkour. [62]

Fig. 09: Robot Atlas [41]

2.1.3.4 Zoomórficos

Los robots que tiene esta arquitectura son los basados en seres vivos. Dentro de estos se pueden encontrar 2 tipos:

- Los no caminadores, los cuales están poco evolucionados, se suelen basar en varias piezas conectadas entre sí, las cuales son capaces de rotar entre ellas.
- Los robots zoomórficos caminadores son objeto de estudio debido a que los investigadores consideran que en un futuro se pueden hacer vehículos basados en sistemas de desplazamiento multípedos, ya que son capaces de desplazarse en superficies irregulares.

Un ejemplo de este tipo de arquitectura es el robot tratado en el paper *"Design and Simulation of a Cockroach-like Hexapod Robot"* [15] en el cual se busca imitar a una cucaracha (figura 10) como base de un robot hexápodo, analizando sus articulaciones y viendo cómo se podría simular un robot que se moviera de una forma similar.

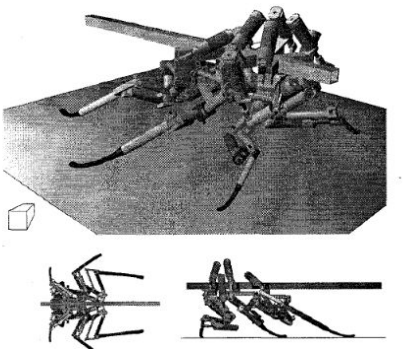


Fig. 10: Esquema de robot cucaracha [15]

2.1.3.5 Híbridos

Los robots híbridos son los que tienen una arquitectura que se sitúa entre una o varias de las que se han mencionado previamente, haciéndolos difíciles de clasificar en una u otra.

Un ejemplo de este tipo de arquitectura es el robot de asistencia Toyota Human Support Robot (figura 11). Este es un robot móvil con un brazo poliarticulado que sirve para asistencia de humanos en sus domicilios. [49]

Fig. 11: Ejemplos de uso del Toyota Human Support Robot [42]

2.2 Paradigmas de la robótica

Dentro de la robótica, se pueden encontrar tres paradigmas que describen los comportamientos de los robots [16]. Para definirlas se deben entender primero las tres primitivas de la robótica SENSE, PLAN y ACT, ya que los paradigmas se describen como las interacciones de estas tres primitivas.

Para definir de una forma más clara las primitivas[19], son las funciones de un robot que cubren un determinado funcionamiento.

- Todas las funciones que traten de conseguir información a través de sensores y devuelvan información útil para el robot, se puede considerar que están en las de categoría SENSE.

- Si es una función que decide una acción en función del conocimiento que tenga el robot, ya sea conocimiento previo o información de los sensores se considera que es de categoría PLAN.

- Si es una función que a través de una orden o de información de sensores ejecuta una acción física en el robot a través de motores o actuadores se considera que se categoriza como ACT.

A continuación se procederá a explicar los tres paradigmas.

2.2.1 Paradigma jerárquico/deliberativo

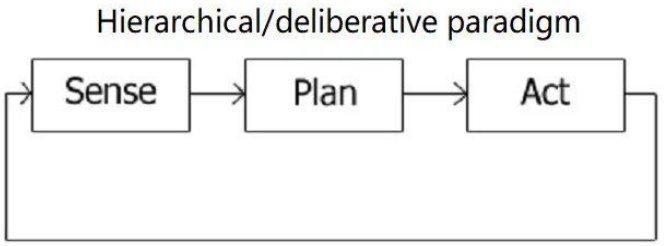


Fig. 12: Esquema del paradigma jerárquico [19]

El paradigma jerárquico o deliberativo es el más antiguo de entre los 3. Estuvo vigente entre 1967 y 1990 y estaba basado en un análisis introspectivo sobre cómo funciona el pensamiento de las personas, "veo algo que quiero, decido como quiero hacer lo que quiero y ejecuto el plan que he pensado previamente para llegar a lo que quiero".

A día de hoy, se sabe que la introspección no es necesariamente una buena idea para analizar la corriente de pensamiento, ya que se sospecha que no todas las acciones se planean premeditadamente sino que se tiene un comportamiento predefinido para realizarlas, así que la base pudo no ser la adecuada.

El modelo se basa en primero sentir con los sensores el mundo exterior, planear un curso de acción y por último ejecutarlo. Después, el curso de acción se repite desde el principio como se puede ver en la figura 12. Otra cosa distintiva de este paradigma sobre otros es que se crea un modelo del mundo que se utiliza para crear los planes y en el que se enrutan las acciones que se planean. Esto supone un problema, ya que los modelos del mundo son difíciles de crear y frágiles, ya que sufren del *frame problem* y también asumen que todo sucede en un mundo cerrado.

2.2.2 Paradigma reactivo

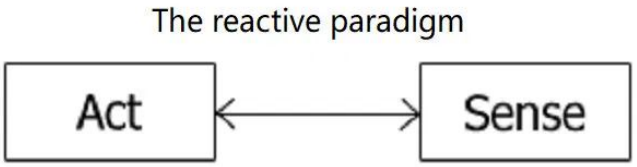


Fig. 13:Esquema del paradigma reactivo [19]

El paradigma reactivo surgió como una respuesta al paradigma jerárquico. Se utilizó sobre todo entre 1988 y 1992 y supuso un gran avance en la robótica, aunque a día de hoy se ha quedado en segundo plano porque el estándar actual son las arquitecturas híbridas.

Este paradigma tuvo dos grandes razones para entrar en auge. Por una parte, se realizó una mayor investigación en biología y psicología cognitiva que llevó a que los modelos de inteligencia artificial se asemejaran más a la realidad y por la otra se incrementó en la potencia de cómputo de los componentes y a su vez se redujo de precio en los mismos.

En este paradigma se realizan modelos SENSE-ACT, eliminando la parte de PLAN, donde cada acción es respaldada directamente por la salida de un sensor como se puede ver en la figura 13. Los robots con arquitecturas reactivas pueden realizar más de una acción porque tienen varios modelos de acción SENSE-ACT llamados comportamientos, los cuales eligen cada uno una acción en función de las entradas de los sensores asociados. Al final se ejecuta una combinación de las salidas de todos los comportamientos, siendo ésta diferente de las salidas de cada comportamiento pero cumpliendo los objetivos de todos ellos.

Aunque los resultados eran buenos en algunos robots, no se podían generalizar para todos, reflejando de alguna forma el problema del estudio de *Skinner* del entrenamiento

con refuerzo en animales, que si bien para algunos animales era útil, no era capaz de explicar la inteligencia humana como tal.

2.2.3 Paradigma híbrido

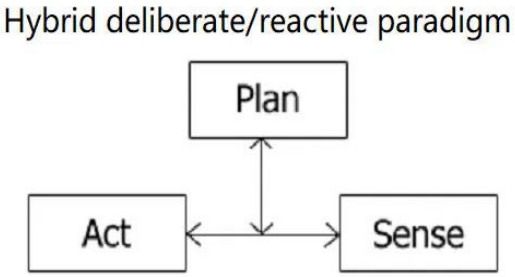


Fig. 14: Esquema del paradigma híbrido [19]

Este paradigma surgió en los años noventa y sigue en uso hasta la actualidad, estando basado en el paradigma reactivo debido a su eficiencia.

En este paradigma se recupera la idea de un planificador pero en vez de decir cómo tiene que actuar el robot, éste divide el objetivo del robot en varias subtareas más pequeñas y asigna comportamientos (del paradigma reactivo) y sensores para solucionarlos de forma eficiente. Esto sucede en dos pasos, primero se planean los comportamientos y sensores (PLAN) y después estos se ejecutan como se puede ver en la figura 14. Adicionalmente, la información de los sensores también se envía al planificador para que realice un modelo del mundo que le permita tomar decisiones sobre los comportamientos. A veces el planificador realiza tareas de escucha para obtener información adicional sobre el mundo de los propios comportamientos.

Cada una de las funciones se ejecutan a diferentes ritmos, ya que el planificador tiene mucha más carga computacional y se ejecuta cada más tiempo (generalmente cada 5 segundos). Por otro lado los comportamientos que son mucho más ligeros

computacionalmente hablando, se ejecutan cada poco tiempo (normalmente cada 1/60 de segundo).

2.3 RoboCup

2.3.1 Introducción

La RoboCup es una competición anual de robótica e inteligencia artificial aplicada a la misma donde se llevan a cabo varias categorías, siendo la más destacable la competición de fútbol de robots, teniendo varias ligas. Además del fútbol, la RoboCup también cubre robótica para niños, robótica dentro de la industria, robótica aplicada a rescate y robótica dentro del domicilio. [18]

Esta competición surgió con el objetivo de que en 2050 un equipo de jugadores robots de fútbol sean capaces de ganar al campeón del mundo del fútbol del momento. Esto puede parecer un objetivo bastante amplio pero en palabras de los propios creadores, buscan un objetivo muy ambicioso como objetivo final para que pueda tener un impacto real en la sociedad y los avances que se consigan en pos de este objetivo sirvan para mejorar la industria y la sociedad en general. Por otro lado, también mencionar que para lograr este objetivo, que actualmente es inviable debido a las limitaciones técnicas, se crean varios objetivos más pequeños que motivan a la gente a seguir avanzando en el campo, como por ejemplo, ser capaces de crear un equipo de fútbol de robots que sea capaz de jugar dada una serie de reglas modificadas.

A su vez, la RoboCup se utiliza para trabajar aplicando algoritmos e inteligencia artificial aplicado a un problema estándar, siendo este el siguiente paso al ajedrez en este ámbito, ya que las características del dominio son mucho más amplias como se puede ver en la tabla 1 y dominarlas supone un avance significativo.[18]

	Ajedrez	RoboCup
Ambiente	Estático	Dinámico
Cambio de estado	Por turnos	A tiempo real
Accesibilidad a la información	Completa	Incompleta
Lecturas de sensores	Simbólica	No simbólica
Control	Central	Distribuido

Tabla 1: Diferencias entre el dominio del ajedrez y el dominio de la RoboCup [17]

Esto da paso a muchos más campos de estudio como puedan ser: fusión de sensores en tiempo real, comportamiento reactivo, adquisición de estrategias, aprendizaje, realización de planes en tiempo real, sistemas multiagentes, reconocimiento de contexto, visión, toma de decisiones estratégicas, control de motores y control de robots inteligente.

Finalmente, mencionar que las reglas son cambiadas anualmente para realizar avances con respecto al objetivo final, evitando así que haya escenarios demasiado artificiales. Estas revisiones de las reglas se planifican en reuniones que suceden cada 5 o 10 años, en las cuales se plantean los avances que deberían haber en función de la situación actual.

2.3.2 Categorías

La RoboCup se estructura en 5 categorías diferentes, cada una con varias ligas que, dentro de la categoría, plantean escenarios y normativas diferentes entre sí, pero sin salirse de la propia categoría.
A continuación se explicará brevemente cada una de ellas.

2.3.2.1 RoboCupSoccer

Esta categoría es la competición principal de la RoboCup en la cual se juega al fútbol. En esta competición el foco de atención de los investigadores reside sobre sistemas cooperativos de varios robots y sistemas multiagentes en ambientes conflictivos que cambian de forma dinámica. En esta categoría los robots deben ser completamente autónomos.

Esta categoría tiene las siguientes ligas:

- **Humanoide**. En esta competición se usan diferentes robots autónomos con forma humana y sensores que simulan la percepción humana (figura 15). Esto último es lo que la diferencia de otras ligas, donde se utilizan robots humanoides pero con otros sensores, haciendo la tarea de percibir el mundo y entender su alrededor más difícil.

  En esta liga se tratan las siguientes dificultades: andar de forma dinámica, correr, patear el balón manteniendo el equilibrio, la percepción y localización de la bola, de otros jugadores, de los límites del campo y la localización del propio robot dentro del campo y por último se trata el trabajo en equipo.

Fig. 15:Categoría de fútbol humanoide en la RoboCupSoccer [43]

- **Plataforma estándar.** Esta competición es la que se utiliza como problema estándar, ya que todo el mundo debe usar el robot NAO de SoftBanks Robotics (figura 16). En esta competición los bots son independientes entre sí y deben comunicarse entre ellos para jugar en equipo. El campo es verde y las líneas del campo y las porterías son blancas. Por último el balón es un balón realista blanco con manchas negras.

  La gama de colores y la falta de referencias extras convierten la situación en una muy complicada que permite que se realicen avances todos los años.

Fig. 16: Categoría de fútbol estándar en la RoboCupSoccer [44]

- **Tamaño medio.** En esta competición cinco robots completamente autónomos juegan con un balón de tamaño regular. Los equipos pueden crear sus propios robots pero los sensores deben ser *on-board* y tienen limitación de tamaño y peso (figura 17).

  El foco de investigación de esta liga reside en los diseños de los robots, el control de los mismos y la cooperación de sistemas multiagentes en cuanto a planificación y percepción del escenario.

Fig. 17: Categoría de fútbol tamaño medio en la RoboCupSoccer [45]

- **Tamaño pequeño**. Esta liga es la más antigua de todas, se centra en el problema de la cooperación de sistemas multi-agente en ambientes dinámicos con un sistema distribuido. En la competición, al igual que en la anterior, existen limitaciones en cuanto al tamaño de los robots y la pelota es una de golf naranja sobre un campo verde (figura 18).

  Los robots son controlados desde ordenadores fuera del campo, que procesan la información y el control de los robots y se envía a través de transmisores y receptores de ondas de radio comerciales.

  La información del campo se consigue a través de 4 cámaras que cuelgan de barras a cuatro metros sobre el mismo.

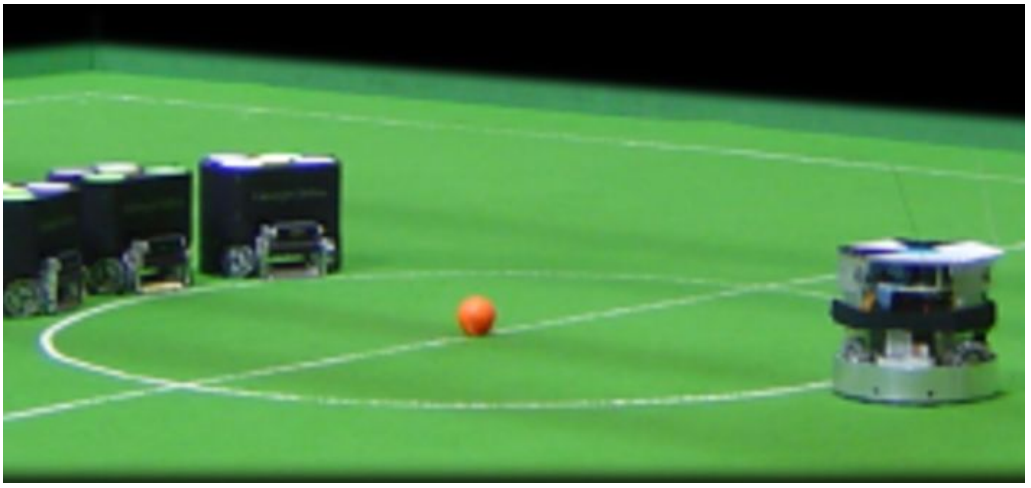


Fig. 18: Categoría de fútbol tamaño pequeño en la RoboCupSoccer [46]

- **Simulación.** En esta liga, centrada puramente en inteligencia artificial y juego en equipo, se realizan simulaciones tanto 2D como 3D (figura 19).

  En la 2D se juega en un servidor que tiene toda la información de lo que sucede y a los agentes se les da una cantidad de información limitada y ruidosa. Estos deben decidir cómo actuar (moverse o patear la pelota) en función de ella.

  En la 3D se simula un ambiente 3D con robots humanoides. Esto centra el objetivo de esta plataforma en controlar comportamientos básicos como andar, patear, girar y ponerse de pie.

Fig. 19: Categoría de fútbol simulación en la RoboCupSoccer [47]

2.3.2.2 RoboCup@Home

Esta categoría consiste en el desarrollo de robots de asistencia con el objetivo de que tengan bastante relevancia en las tareas domésticas, siendo esta la mayor competencia anual de robots de servicio. Esta competición se considera un *benchmark* sobre el desempeño de los robots de servicio en un ambiente realista no estandarizado. Se centra en la interacción con el humano, la manipulación de objetos, la integración de los comportamientos, el mapeo del ambiente y el reconocimiento de elementos en ambientes con iluminación realista.

Esta categoría tiene las siguientes ligas:

- **Plataforma abierta**. En esta categoría se puede participar usando cualquier robot propio (figura 20a).
- **Plataforma doméstica estándar**. En esta categoría se utiliza el Toyota Human Support Robot (HSR) para competir (figura 20b).
- **Plataforma social estándar**. En esta categoría se utiliza el Softbank Robotics Pepper para competir (figura 20c).

Fig. 20a: Categoría de plataforma abierta de RoboCup@Home. [48]

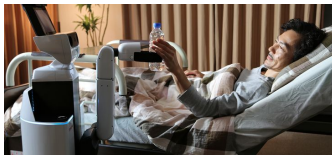


Fig. 20b: Categoría doméstica estándar de RoboCup@Home. [49]

Fig. 20c: Categoría social estándar de RoboCup@Home. [50]

2.3.2.3 RoboCupIndustrial

En esta categoría se recogen dos competiciones que tratan el uso de robots en ambientes de trabajo e industriales así como la aplicación de robots a logística.

Esta categoría tiene las siguientes ligas:

- **RoboCup@Work**. Esta es la liga más nueva de la RoboCup y a través de ideas de otras competiciones de la RoboCup cubre los retos que suponen el uso de robots en ambientes industriales y servicios aplicados a ambientes de trabajo

- **Logistics**. Esta liga se desarrolla en un ambiente de trabajo similar a una fábrica, donde se prueban robots que manufacturen, refinen, monten o modifiquen algo que termine convirtiéndose en un producto terminado (figura 21).

Fig. 21: Categoría de logística de RoboCup@Work. [51]

2.3.2.4 RoboCupJunior

Esta categoría está orientada a animar a jóvenes que aún no puedan participar en las competiciones de adultos a interesarse en la robótica e inteligencia artificial, dándoles la oportunidad de viajar a otros países y conocer a gente con intereses comunes. Así pueden tener un acercamiento a la robótica e inteligencia artificial desde un enfoque más cooperativo aprendiendo a trabajar en equipo, a diferencia de otros enfoques donde cada joven tenga un ordenador y no interactúe con nadie más como se suele ver hoy en día.

Esta categoría tiene las siguientes ligas:

- **Rescate**. En esta liga los robots identifican víctimas en escenarios de desastres creados con diferente complejidad, planificando y recorriendo caminos entre los obstáculos sobre un terreno desigual.

- **Fútbol**. Equipos de dos robots autónomos compiten en un campo cerrado con marcas y una pelota que emite luz (figura 22).

- **Onstage**: Robots y personas salen a un escenario a realizar una escena, interactuando entre sí.

Fig. 22: Categoría de fútbol en la RoboCupJunior. [52]

2.3.2.5 RoboCupRescue

En esta categoría se busca mejorar y trabajar en los retos relacionados con sistemas de rescate y búsqueda.
Se entrará en más detalle en el apartado 2.3.3.

2.3.3 Rescate

La categoría de rescate de la Robocup [20] se centra en mejorar tanto los sistemas reales de búsqueda y rescate en situaciones que representen catástrofes o desastres naturales, como en sistemas simulados e incluso las propias herramientas de simulación.

Esta categoría está formada por dos ligas, una en la que se compite con robots reales y la otra que se compite con simulaciones. Está ultima se descompone en 3 sub ligas, simulación de agentes, simulación de robots y simuladores.

2.3.3.1 Robot

Fig. 23: Ejemplo de robot de rescate. [53]

Está liga se centra, como se ha mencionado previamente, en una competición anual centrada en la creación de robots (figura 23) con usos para búsqueda y rescate y para desactivación de bombas. Con esto se consigue incrementar la preocupación de la gente sobre los problemas que suponen estas tareas, ofrecer un análisis objetivo de cómo de buenos son los robots en este tipo de tareas y animar a los investigadores a colaborar más entre sí, llegando a crear mejores robots con los puntos fuertes de cada equipo de investigación.

Con esta competición se busca mejorar en las siguientes habilidades de los robots:

1. Lidiar con estructuras peligrosas o destruidas.
2. Encontrar víctimas y asegurarse de su estado.

3. Crear mapeos prácticos de lugares a través de sensores.
4. Establecer comunicación con víctimas.
5. Entregar medicinas, comida o fluidos.
6. Poner sensores para monitorizar peligros.
7. Encontrar los mejores caminos hacia víctimas.
8. Ofrecer apoyos estructurales a vigas.

La competición consta de una fase preliminar donde se evalúan las capacidades de un robot y después los robots que pasan esta prueba se ponen en un escenario simulado donde se prueban las capacidades del robot. Por último, la final consta de una prueba donde se evaluará la capacidad de búsqueda e identificación de víctimas simuladas en un laberinto.

2.3.3.2 Simulación

[21]Esta liga cubre los problemas planteados en general aplicados a simulación. Esto se enfoca de dos formas diferentes, simulación de agentes y simulación de robots. Además de esto existe una tercera competición que apunta a mejorar las herramientas de simulación a través de las cuales se realizan las otras competiciones.

Las 3 sub ligas son:

- **Agent Simulation.** Esta sub liga apunta a simular y evaluar sistemas multiagentes en diferentes mapas de la plataforma de RoboCup Rescue Agent Simulation (RCRS). Aquí se busca evaluar la efectividad de diferentes servicios (ambulancias, policía y bomberos) a la hora de rescatar civiles y extinguir fuegos en una ciudad donde acaba de suceder un terremoto. En cada una de las rondas se evalúan diferentes situaciones a través de diferentes mapas y se puntúa cada grupo de agentes por cada ronda (ejemplo en la figura 24). [22]

El objetivo principal de esta competición subyace en mejorar el trabajo en equipo de los servicios de emergencia en situaciones extremas.

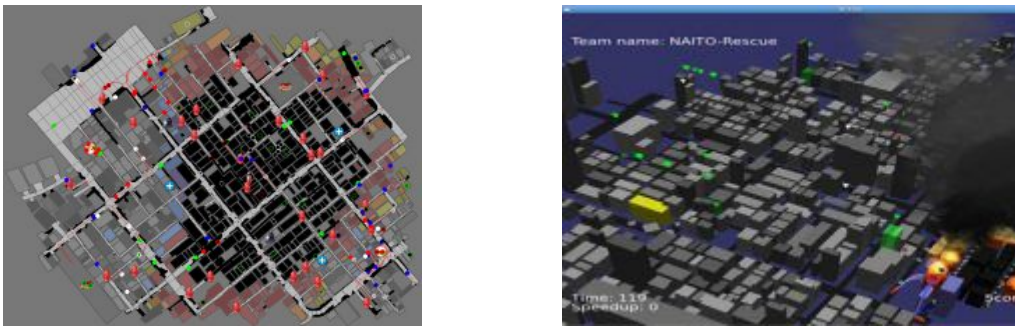


Fig. 24: Ejemplos de agentes actuando sobre un mapa. [54] [55]

- **Robot Simulation.** En esta subliga se realiza una tarea similar a la liga de robots, pero en un entorno simulado como se puede observar en la figura 25.[23]

Fig. 25: Ejemplo de robots actuando sobre un mapa. [56]

- **Infrastructure.** Esta sub liga es una competición en la que se presentan herramientas ya conocidas y se busca realizar mejoras sobre ellas, cada aspecto que se plantea se evalúa con una puntuación y finalmente gana el equipo que

mayor puntuación tenga. Las puntuaciones las dan los participantes en las otras sub ligas de simulación y expertos. [24]

Todos los participantes deben dar el código fuente del programa como *open-source* antes de la competición, además de un manual completo y preparar *scripts* para su instalación y uso, sino serán descalificados.

2.3.3.3 Robots hexapodos en rescate

Uno de los posibles modelos de robots usados en las tareas de rescate son los robots hexápodos.

Los robots hexápodos tienen una gran ventaja a la hora de recorrer terrenos irregulares porque al tener seis puntos de apoyo relativamente extendidos son capaces de que su centro de gravedad permanezca en una posición estable aun en pendientes inclinadas haciéndolos ideales para estas.

Otra de las ventajas que tienen este tipo de robots es que al tener un método de locomoción basado en patas en vez de ruedas son capaces de atravesar terreno dificultoso apoyándose únicamente en las zonas estables y de sortear obstáculos levantando las patas. Otra de las ventajas que aporta este método de locomoción es que permite ajustar la altura a la que va el cuerpo del robot, de forma que se pueda adaptar mejor al terreno.

2.4 Simuladores

Un simulador de robótica es un programa que tiene como objetivo ofrecer una versión virtual de uno o varios robots para poder trabajar con esta sin necesidad del propio robot, abaratando así costes y permitiendo trabajar a velocidades distintas de las que se podrían con la versión física del robot [25].

Como ya se ha mencionado previamente, los simuladores permiten realizar imitaciones de cómo se podría comportar un robot en una determinada situación planteada a priori. Hay ejemplos de esto mismo en el Apartado 2.3.3.2, donde se habla de diferentes situaciones catastróficas generadas previamente. Estos "mundos" generados suelen ser varios objetos rígidos y fuentes de luz colocados de una determinada forma y los robots que se simulen deberán ser capaces de interactuar con ellos a través de actuadores o de sus sensores.

Los simuladores suelen cumplir una serie de características:

- Ser capaces de realizar prototipos de robots desde el propio simulador.
- Tener motores de físicas para poder generar un movimiento realista (normalmente ODE o Physx)
- Tener una representación realista de modelos 3D. Para esto se pueden utilizar herramientas estándar de modelado o programas de terceros.
- Simular robots de forma dinámica a través de *scripts* en diferentes lenguajes de programación.

2.4.1 Webots

Webots [26] es un simulador de software libre el cual pasó previamente por una etapa de pago. Este simulador tiene bastante a su favor si se quisiera realizar alguna tarea con él, ya que cumple con las siguientes características [25]:

- Su lenguaje principal es C++ pero admite programación en C, C++, Python, Java, Matlab y ROS y su interfaz de usuario es una GUI.

- Tiene evasión dinámica de colisiones, efectores finales relativos, programación fuera de línea y control de transmisión de hardware en tiempo real.

- Tiene una API de documentación, un foro, un manual de usuarios, un rastreador de problemas y una wiki por si fuera necesaria asistencia.

- Tiene robots de las siguientes familias: robots móviles de tierra, robots aéreos, robots submarinos, brazos robóticos, manos robóticas, robots humanoides y más.

- Tiene los siguientes sensores: Odometría, IMU, colisión, GPS, cámaras monoculares, cámaras estéreo, cámaras de profundidad, cámaras omnidireccionales, escáneres láser 2D y escáneres láser 3D.

En cuanto a la interfaz de usuario, podemos ver lo siguiente:

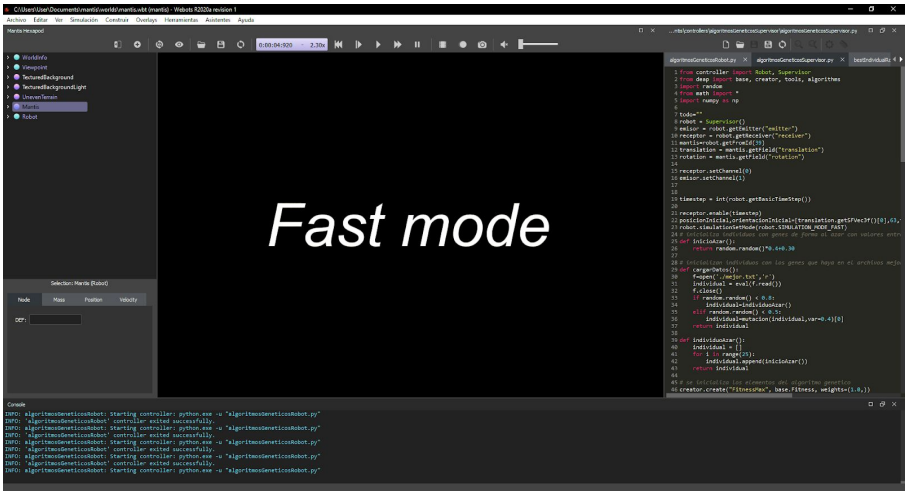


Fig. 26: Layout del simulador Webots

En la figura 26 se puede ver una barra de herramientas con los controles de la simulación, así como opciones para modificar el sonido de la misma o grabarla. A la izquierda podemos encontrar una lista con los elementos del mundo. Bajo esto podemos ver una pequeña ventana donde poder variar los parámetros de los elementos mencionados previamente. En el centro existe una ventana donde se muestra la simulación del robot. A la derecha un editor de texto desde el que se puede trabajar en los controladores y en la parte inferior una consola.

Una de las utilidades de este simulador es que permite, a través de su librería propia, controlar todo el sistema mediante su clase supervisor. Esto permite modificar posiciones, rotaciones, velocidades o cualquiera de los campos de los elementos de la simulación y también reiniciar la simulación o los controladores de la misma, cambiar su velocidad o acabarla, haciéndolo idóneo para aplicaciones de inteligencia artificial que requieran aprendizaje.

Por último, mencionar que la lista de robots con los que se puede trabajar está en su página web [27], la de actuadores en la que corresponde [28] y los sensores en la suya correspondiente [28].

2.4.2 Mantis

Uno de los robots posibles de controlar en Webots es el robot Mantis [63]. Este robot creado por Micro Magic es un robot hexápodo, el cual tiene 3 articulaciones por cada pata, siendo estas motores que permiten un movimiento entre $-\pi$ radianes y π radianes, representando así un movimiento de 360 grados.

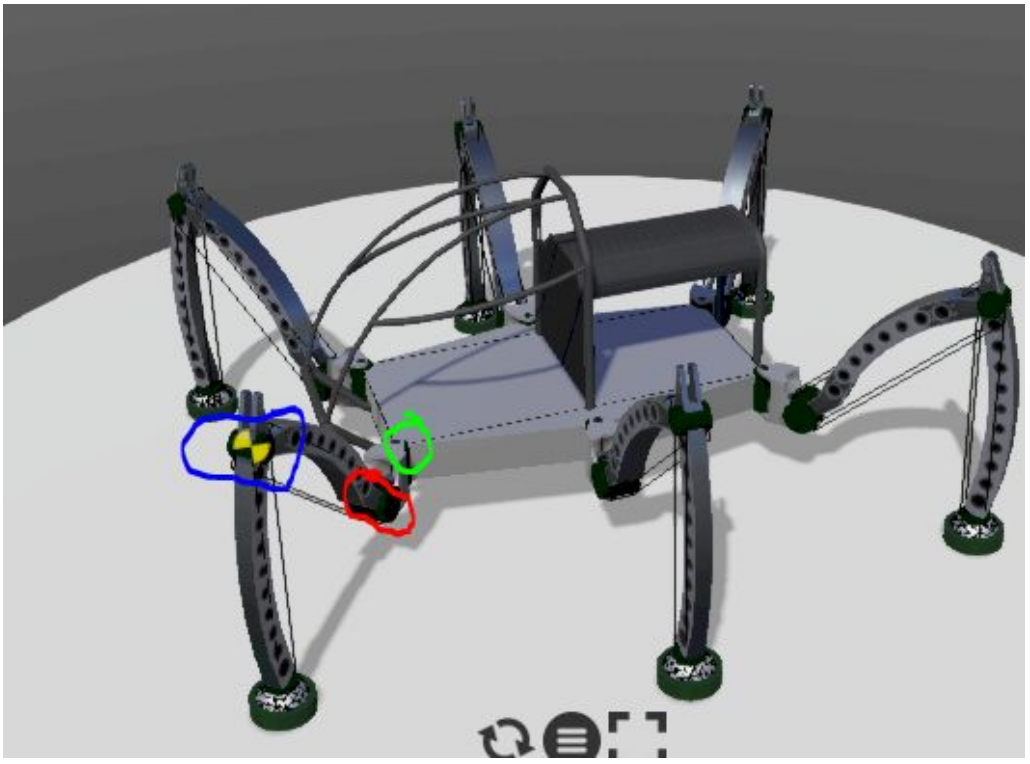


Fig. 27: Robot Mantis

Como se puede ver en la figura 27 las 3 articulaciones que tiene son:

- **Base.** Es la articulación, marcada en verde, que permite la rotación horizontal de la pata.
- **Hombro**. Es la articulación, marcada en rojo, que permite el desplazamiento vertical de la pata.

- **Rodilla**. Es otra articulación, marcada en azul, que permite el desplazamiento vertical de la segunda mitad de la pata, dando así más libertad a las patas del robot.

Además de esto, el robot tiene variables que indican: su posición en los 3 ejes, su rotación, su nombre, el controlador que va a usar, si es supervisor y si funciona de forma síncrona. Por otro lado tiene una variable llamada espacios de ampliación, donde se le pueden añadir elementos extra, como nuevos sensores o actuadores.

2.5 Algoritmos genéticos

Entre las diferentes técnicas de inteligencia artificial cubiertas en el campo de la computación, los algoritmos genéticos son una de ellas, enfocadas a la optimización de sistemas. Usualmente se suelen utilizar cuando se tiene una solución o una definición de la solución de un problema pero se quiere hallar la que lo resuelve de forma óptima. Por eso, algunas de sus aplicaciones más comunes son la optimización de redes de neuronas y la de los comportamiento de robots [32].

Los algoritmos genéticos funcionan basándose en la evolución natural, es decir, dado un conjunto de individuos en una determinada situación solo sobreviven y se reproducen los que están más adaptados al medio, teniendo descendencia con genes compartidos por individuos bien adaptados, los cuales pueden sufrir mutaciones que los hagan mejores o peores en la tarea de adaptación, sometiendo a los nuevos individuos a la misma situación que sus padres. En esta técnica se hace algo similar: la adaptación al medio se define como una función que da valores a los individuos que mejor estén realizando la tarea requerida y después de una determinada cantidad de generaciones, el individuo más adaptado resultante de los cruces y mutaciones que suceden en éstas es el resultado final como se puede ver en la figura 28.

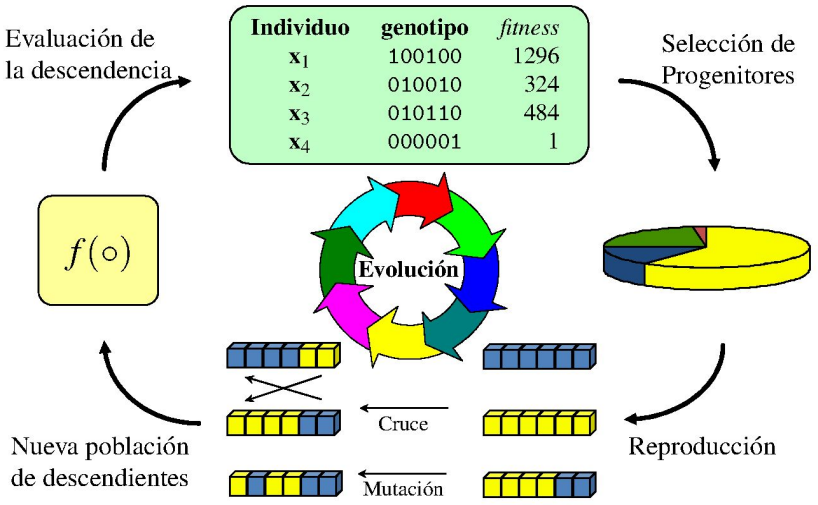


Fig. 28: Estructura de un algoritmo genético [64]

Un algoritmo genético está compuesto por una población de individuos y unos operadores que se aplican a los mismos. Los elementos son:

- Un **individuo** es una lista de diferentes genes los cuales representan la información de la solución. Estos genes suelen ser valores de los parámetros de la solución pero pueden tomar otras formas.

- Los **operadores** son las funciones encargadas de realizar la evolución. Entre estos podemos encontrar los operadores:
    - Mutación, los cuales definen las modificaciones que puedan tener los individuos.
    - Sobrecruzamiento, los cuales intercambian genes entre individuos.
    - Selección, los cuales deciden en función de los valores de la función de fitness qué individuos forman parte de la población de la siguiente generación.

- La **función de fitness** define a través de una función matemática cómo de bueno ha sido el desempeño del individuo a la hora de realizar la tarea.

Capítulo 3: Análisis y diseño de un sistema de marcha de un robot hexápodo

3.1 Software utilizado

A continuación se procederá a realizar una explicación del software utilizado, así como el razonamiento detrás del mismo:

- **El Sistema Operativo** empleado para la realización de este trabajo ha sido Windows 10 Home de 64 bits. Esto es debido a que es el Sistema Operativo más implementado a nivel mundial y en consecuencia tiene un gran soporte en caso de que haya cualquier tipo de fallo. El resto de softwares utilizados tienen una buena integración en este sistema operativo y ya se encontraba instalado y configurados todos sus drivers en el terminal desde el que se ha realizado la implementación y el entrenamiento del algoritmo genético.

- **El Simulador** en el que se ha implementado el trabajo ha sido Webots, que como ya se ha mencionado previamente es software libre, tiene mucha y buena información a disposición de sus usuarios (API, manual de usuario y un foro activo), además de una buena integración de los controladores dentro de la interfaz del propio simulador a través de un editor de texto, así como un fácil acceso a las funcionalidades del robot usando una librería. Por último, mencionar la existencia de un robot hexápodo en el repertorio de robots del simulador.

- **El lenguaje** en el que se ha realizado la implementación del controlador ha sido Python, ya que es un lenguaje muy extendido, tiene un buen soporte por parte de la comunidad además de muchas librerías gratuitas y una buena compatibilidad con el simulador.

- **Las librerías** utilizadas para la realización del trabajo son las siguientes:

    - *Controller*, ofrecida por el propio simulador, permite controlar cada una de las partes del robot así como contactar al robot con el supervisor a través de emisores y receptores y modificar y moldear la simulación gracias a la clase de Supervisor.

    - *Random*, utilizada para todas las operaciones probabilísticas, así como para la creación y mutación de los individuos.

    - *Math,* para las operaciones trigonométricas como senos y otras como valor absoluto, raíces y también para tener acceso con cierta exactitud al número π.

    - *Numpy,* para extraer información estadística del fitness de la población en cada generación.

    - *Deap*, para tener soporte para algoritmos genéticos, permitiendo trabajar con poblaciones e individuos a través de clases, crear o utilizar operadores genéticos y sacar estadísticas del entrenamiento.

3.2 Análisis de requisitos

El análisis de requisitos se va a dividir en tres partes. Por un lado se mencionan los requisitos funcionales del controlador asociado al robot, después se enumeran los requisitos funcionales del supervisor y finalmente se enuncian los requisitos no funcionales de la implementación en sí.

3.2.1 Requisitos funcionales del robot

- RFR1: El controlador inicializa el robot y sus componentes.

- RFR2: El controlador debe recibir la información del individuo.

- RFR3: El controlador debe colocar al robot en una posición de *standby* derivada de la información del individuo.

- RFR4: El controlador debe asegurar que el robot esté estable antes de iniciar el proceso de marcha.

- RFR5: El controlador debe ser capaz de, dados unos parámetros, realizar un proceso de marcha que se adecue a ellos.

- RFR6: El controlador debe ser capaz de conseguir la información referente a la activación de los sensores necesaria para la función de fitness y la aplicación de penalizaciones en caso de requerirse.

- RFR7: El controlador debe enviar la información del desempeño del individuo una vez realizado el experimento en el tiempo establecido.

3.2.2 Requisitos funcionales del supervisor

- RFS1: El controlador debe inicializar todos los componentes del supervisor.
- RFS2: El controlador debe configurar la velocidad de la simulación rápido.
- RFS3: El controlador debe inicializar la población del algoritmo genético.
- RFS4: El controlador debe ser capaz de aplicar los diferentes operadores genéticos.
- RFS5: El controlador debe ser capaz de evaluar el fitness de un individuo.
- RFS6: El controlador debe ser capaz de seleccionar una población nueva dada una población sobre la que se han ejecutado operadores genéticos.
- RFS7: El controlador debe ser capaz de enviar la información correspondiente al controlador del robot.
- RFS8: El controlador debe ser capaz de recoger información sobre el fitness de la población.
- RFS9: El controlador debe tener algún mecanismo para asegurar elitismo.
- RFS10: El controlador debe almacenar en un documento de salida la información del fitness de la población.

- RFS11: El controlador debe almacenar el mejor individuo tras la ejecución del algoritmo genético.

3.2.3 Requisitos no funcionales

- RNF1: La comunicación entre controladores se realizará usando los elementos emisor y receptor incorporados en el simulador.

- RNF2: Todas las simulaciones se realizarán en el simulador Webots.

- RNF3: Los controladores se programarán en python.

- RNF4: La información será guardada en ficheros .txt.

Capítulo 4: Implementación del sistema de marcha de un robot hexápodo

En este capítulo se tratarán todos los temas relacionados con la implementación del trabajo, cómo se ha desarrollado, qué resultados se han obtenido y un análisis de los mismos.

4.2 Primeras aproximaciones

En este apartado se explicarán los primeros acercamientos a través de los cuales se optó por desarrollar el proyecto, qué resultados dieron y por qué no se siguió con ellos.

En una primera instancia se optó por realizar tanto el movimiento del robot como el algoritmo genético y las funciones del supervisor que lo optimizaron en un mismo controlador. Siguiendo este modelo se probaron varios experimentos los cuales no resultaron satisfactorios.

En el primer enfoque se optó por que el controlador reaccionara a las entradas de los sensores de las patas. Esto consiste en realizar una matriz de pesos para cada motor en la cual se representará un peso por cada uno de los valores de activación de los sensores [-1,1] de forma que cada $motor_j$ debería terminar en una posición determinada por la función:

$$PosMotor_j = \sum_{i=1}^{6} peso_{ij} * sensor_i$$

Este enfoque tenía el problema de que el robot no tenía un comportamiento cíclico como se podría esperar, sino que cada vez que las patas tocaban el suelo las levantaba y al levantarlas las volvía a bajar debido a que el sensor dejaban de tener ninguna entrada, haciendo que no se desplazase, solo se quedase en el sitio realizando pequeños saltos.

A raíz del fallo observado en el desarrollo del enfoque anterior se optó por tomar un camino más parecido al del controlador de ejemplo que proporciona Webots, donde se utiliza una función seno para definir el comportamiento de los motores. Siguiendo este modelo se optó por definir una función similar y optimizar los parámetros necesarios. Al final el resultado fue:

$$PosMotor_j = amplitudd_j * sin(2 * \pi * frecuencia * tiempoActual + fase_j) + desplazamiento_j$$

Siendo amplitud, fase y desplazamiento los parámetros a optimizar.

4.3 Un controlador

A continuación se procederá a explicar los primeros experimentos que se realizaron una vez se adoptó esta función de definir la posición de los motores, la cual ha continuado usándose hasta el final del proyecto.

1. En el primer experimento se optó por realizar 500 generaciones con 50 individuos. En este experimento los individuos estaban definidos como una lista de 540 bits, los cuales a la hora de evaluar el individuo se separaban en 3 listas de 18 números decimales a los cuales les correspondía 10 bits a cada uno. Estos valores después se normalizaron y se multiplicaban por los valores máximos que podían tener la amplitud, la fase y el desplazamiento de los motores y se introducían en la fórmula.

   Los operadores eran: cruce en 2 puntos para el *crossover,* con una probabilidad del 80% de ejecutarse y cambio de bits para la mutación, con una probabilidad del 20% de suceder y de un 10% de mutar cada bit, torneo para la selección, siendo el tamaño del torneo de 15 individuos y para la evaluación, la función de

fitness correspondía a la distancia que se ha desplazado el robot en cualquier dirección multiplicado por el número de sensores que han estado en contacto con el suelo menos 3 por cada instante de tiempo. Esto último es para asegurar que tiene al menos 3 patas en el suelo en todo momento, que es la forma en la que se considera que un robot hexápodo sea estable.

En la tabla 2 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	500
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Bit
	Tamaño	540 Genes
Mutacion	Prob. Individuo	20 %
	Prob. Gen	10 %
Cruce	Prob. Individuo	80 %
	Prob. Gen	No aplica
Selección	Tamaño torneo	15 Individuos

Tabla 2: Datos del experimento uno de un controlador

Este enfoque dio los siguientes resultados:

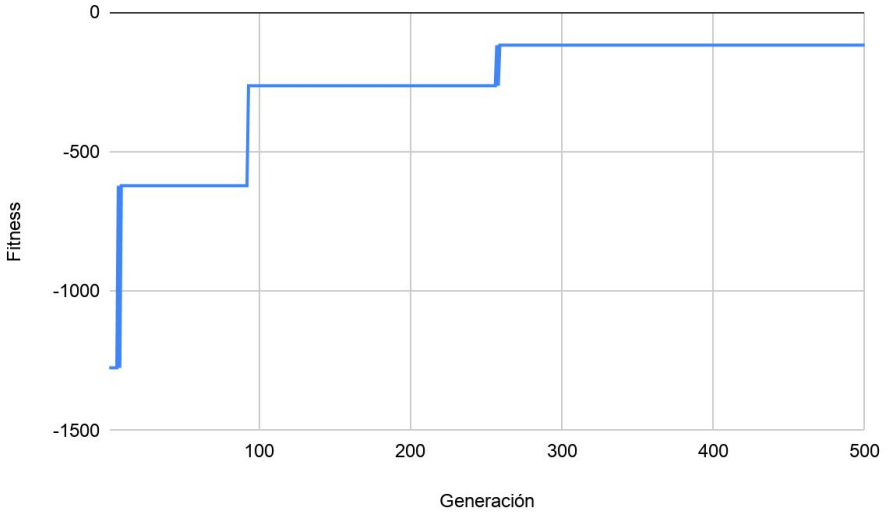


Fig. 29: Fitness del primer experimento.

Como se puede observar, si bien va mejorando a lo largo del tiempo, mejora 3 veces, entre las diferentes mejoras pasan en algunos casos más de 150 generaciones. Además podemos ver que a partir de la generación 259 se queda estancado y no vuelve a haber ninguna mejora. Esto se debe a que la definición del individuo era demasiado grande y con valores binarios, haciendo que tenga la misma probabilidad de modificar un bit que corresponda al valor 2^9 que uno que corresponda al valor 2^0, de forma que en vez de variar poco a poco los valores fuera dando saltos por todos los valores posibles.

2. El segundo experimento tuvo unas dimensiones similares al primero pero se cambió la forma que tenía el individuo, en este caso se definió como una lista de 54 valores entre 1 y 1000 los cuales se utilizaban normalizados al igual que en el caso anterior.

En este caso se realizó un cambio en los operadores ya que en el caso anterior se utilizaban para valores binarios y ahora ya no eran aplicables. En cuanto al *crossover* se utilizó un operador de cruce uniforme el cual tenía un 80% de probabilidades de aplicarse y una vez aplicado un 20% de cruzar los genes. De cara al operador de mutación se utilizó uno definido en el código, el cual para cada gen dada una probabilidad introducida por parámetro sumaba o restaba al gen un valor definido aleatoriamente de forma que el resultado estuviese comprendido entre 0 y un valor máximo (introducido por parámetro), modificandolo más o menos según se quisiera y definiendo unos límites los cuales nunca se superarían. Este último operador se utilizaba con una probabilidad que variaba entre un 5% al principio, hasta un 15% en la última generación y los parámetros introducidos eran un 10% de variabilidad, límites entre 0 y 1000 y una probabilidad de mutar cada gen del 10%. La operación de selección se cambió para seleccionar el mejor. La función de fitness no varió del experimento anterior.

En la tabla 3 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	500
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1000]
	Tamaño	54 Genes
Mutacion	Prob. Individuo	5-15 %
	Prob. Gen	10 %
Cruce	Prob. Individuo	80 %
	Prob. Gen	20 %
Selección	Mejor individuo	No aplica

Tabla 3: Datos del experimento dos de un controlador

En este caso los resultados fueron:

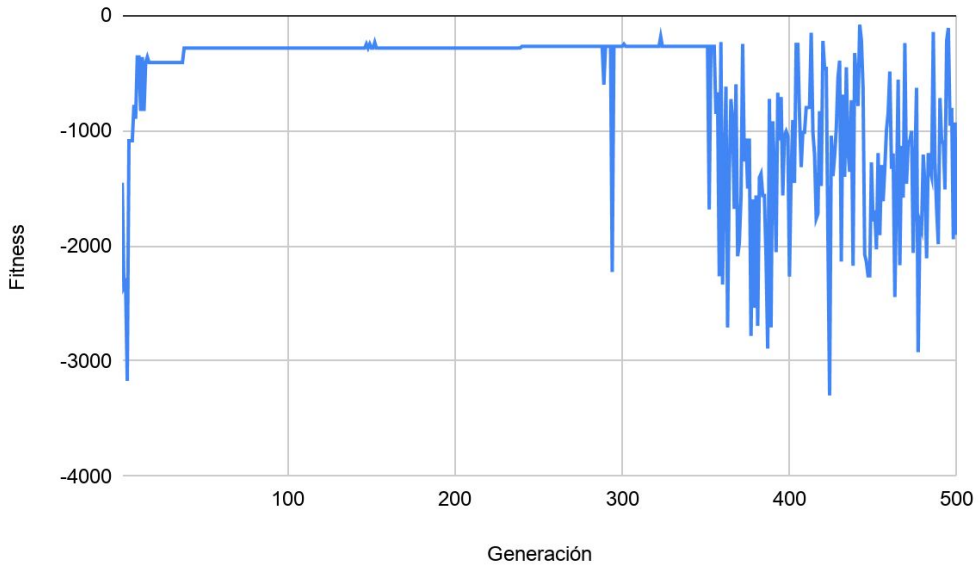


Fig. 30: Fitness del segundo experimento.

Como se puede observar el principio es algo mejor que el de la otra gráfica pero tiene varios picos, esto es debido a que aún no había ningún mecanismo para asegurar el elitismo y ruido que introducía el simulador. El ruido se puede ver también en torno a la generación 150 y a 290. Pero lo que más llama la atención es que a partir de la generación 350 pierde totalmente la consistencia, esto es debido a un error del simulador el cual dejó de obtener la información correcta de los sensores. Este error, como se podrá ver más adelante, es recurrente y ha forzado a trabajar en torno a él para poder evitarlo.

3. En el siguiente experimento las dimensiones se fijaron iguales a las del anterior pero en este caso se cambiaron los valores de los individuos a valores entre 0 y 1 para no necesitar normalizarlos a la hora de trabajar con ellos.

   En cuanto a los operadores, se volvió a usar los del experimento anterior pero con la variación de que los límites establecidos entre 0 y 1000 en la función de

mutación pasaron a ser valores entre 0 y 1 y para definir al individuo se usó la función *random.random()* en lugar de *random.randint()* que se había usado previamente para generar números decimales directamente en vez de números enteros y luego normalizarlos, ganando mayor precisión con ello. La función de fitness y el resto de operadores no variaron.

En la tabla 4 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	500
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	54 Genes
Mutacion	Prob. Individuo	5-15 %
	Prob. Gen	10 %
Cruce	Prob. Individuo	80 %
	Prob. Gen	20 %
Selección	Tamaño torneo	No aplica

Tabla 4: Datos del experimento tres de un controlador

Los resultados obtenidos en este experimento han sido:

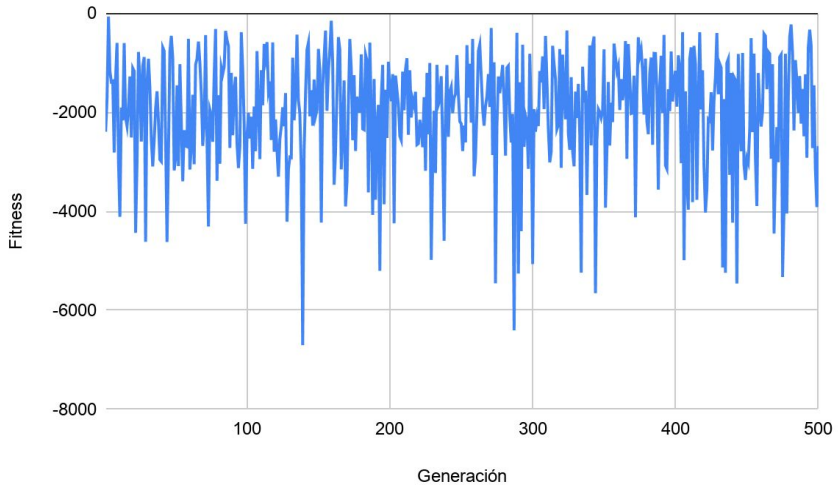


Fig. 31: Fitness del tercer experimento.

Como se ha mencionado previamente los sensores no estaban tomando lecturas correctas al reiniciar la simulación y ya que la función de fitness depende de estos sensores, los resultados de la función eran erróneos y arbitrarios haciendo que no evoluciona correctamente.

4. En el cuarto experimento se repitió el anterior sin ninguna variación con esperanzas de que salieran resultados coherentes, ya que previamente se habían conseguido en el experimento 1 y 2.

Estos fueron los resultados conseguidos:

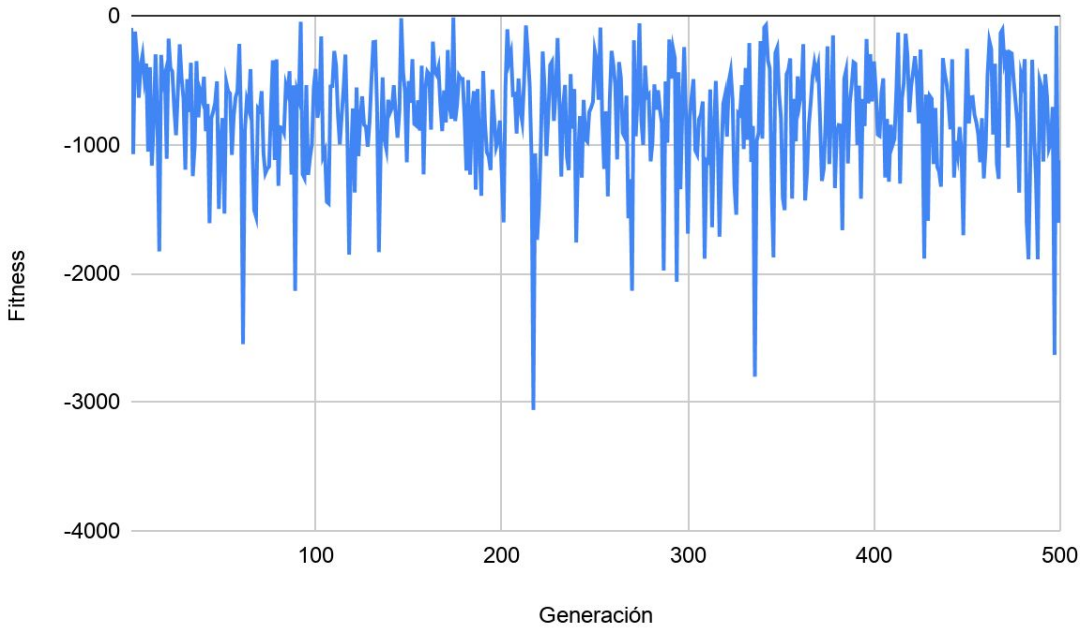


Fig. 32: Fitness del cuarto experimento.

Estos resultados fueron similares a los anteriores. La razón por la cual se siguió haciendo experimentos enfocados de esta forma fue que aún se desconocía el error en los sensores.

5. En el quinto y último experimento de un único controlador, se implementó un sistema que duplicaba los individuos cuando se mutaban y se cruzaban, permitiendo así que quedara únicamente el mejor individuo entre uno mutado y uno cruzado. Este método permitió que existiese elitismo y además se duplicó el tamaño del experimento.

En la tabla 5 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	1000
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	54 Genes
Mutacion	Prob. Individuo	15 %
	Prob. Gen	5 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	No aplica

Tabla 5: Datos del experimento cinco de un controlador

Los resultados aplicando este método fueron los siguientes:

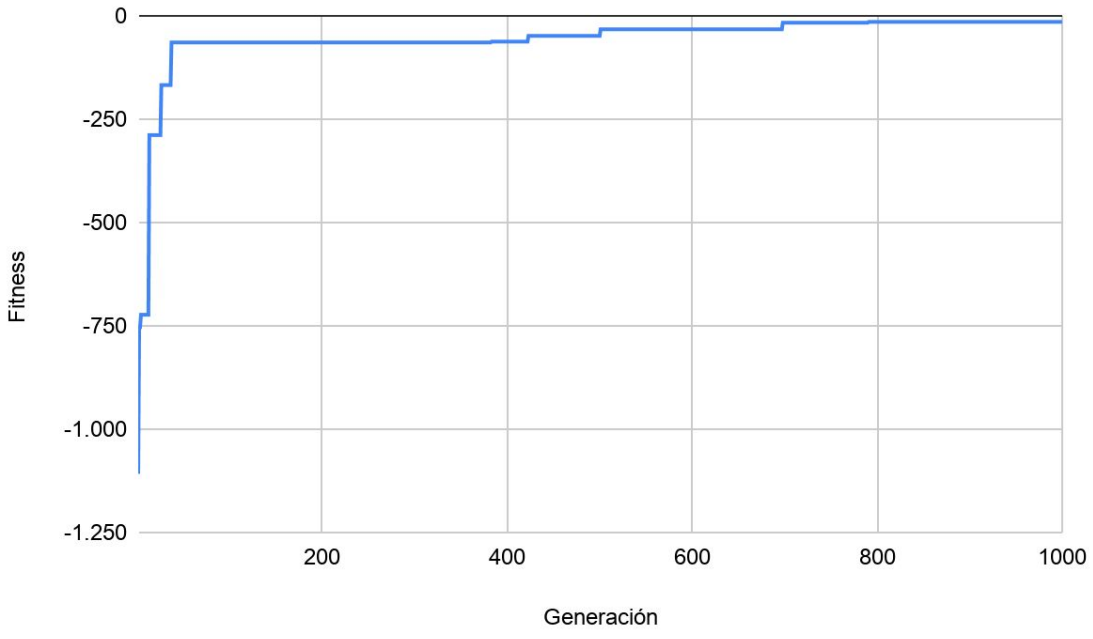


Fig. 33: Fitness del quinto experimento.

Si bien los experimentos son aparentemente mejores y ya se comporta como debería, en este momento fue cuando se descubrió que no estaban captando las señales de los sensores salvo en el primer individuo de la primera generación.

El añadir elitismo dio la sensación de que aprendía, pero simplemente usaba la señal de los sensores del último instante de tiempo del primer individuo de la primera generación en todos los instantes de todo el resto de individuos, siendo un resultado erróneo.

4.4 Dos controladores

Para solucionar el problema de los individuos se decidió separar el control del robot en un controlador y el control de la simulación en otro. Para esto se creó un robot vacío el cual actuaba de supervisor y junto al controlador del robot se comunicaban a través de receptores y emisores.

Ya que la comunicación de los robots es algo más lenta que realizar todo en un controlador, en estos experimentos se han usado tamaños de 100 generaciones para realizarlos y se ha cargado una parte de los resultados en el siguiente experimento para que se mantuviese el avance entre generaciones. Esto implica que la longitud de los diferentes experimentos no será constante ya que cada uno de los diferentes acercamientos o cambios implica varios experimentos, algunos partiendo de los resultados obtenidos en otros de los acercamientos.

Habiendo realizado un análisis exhaustivo sobre los controladores se concluye que el problema de los sensores venía derivado de la función *simulationReset(),* ya que al no reiniciarse el controlador la inicialización de los sensores se quedaba haciendo referencia a un robot anterior y no se reiniciaba. Esto, aunque no debería suceder, se pudo solucionar con la división en dos controladores, ya que se puede utilizar la función *restartController()*, la cual elimina la instancia anterior del controlador y crea una nueva con nuevas referencias a los sensores.

Una posible alternativa que se probó antes de llegar a la utilización de la función *restartController()* fue modificar el parámetro de la posición del robot y su rotación, pero después de varias ejecuciones, debido a un error en las físicas del simulador, las uniones de los sensores del robot se desconectaban, haciendo que los sensores flotaran delante de las patas, cambiando de forma significativa el comportamiento del robot.

Una vez solucionados los problemas con los sensores se procedió a programar los diferentes controladores para la optimización del robot. La estructura de estos es la siguiente:

Robot:

1. Inicialización de los componentes del robot.
2. Recepción de la información del individuo.
3. Carga de los datos en el robot.
4. Ejecución durante el tiempo establecido.

Supervisor:

1. Inicialización de los componentes del supervisor.
2. Definición de los parámetros iniciales de la simulación.
3. Definición de funciones del algoritmo genético.
4. Definición de los parámetros del algoritmo genético.
5. Ejecución del algoritmo genético.

Partiendo de esta estructura los experimentos realizados con dos controladores son los siguientes:

1. El primer conjunto consta de 5 experimentos con una población de 50 individuos y un tamaño de 100 generaciones salvo el cuarto que tiene 150. En estos experimentos se evalúa el desempeño del robot durante 10 segundos y la única variación sobre los operadores mencionados anteriormente es su función de fitness, la cual pasa a ser:

$$suma<0: \quad suma$$
$$suma\geq 0: \quad DistRecorrida$$

   Siendo *suma* la suma de los valores de los sensores menos tres en todos los instantes de tiempo y *DistRecorrida* la distancia recorrida en horizontal desde la

posición inicial. Además, cabe mencionar que, la forma de extender los resultados entre experimentos es una función la cual crea un 90% de individuos nuevos, un 5% iguales al mejor del anterior y un 5% partiendo del anterior pero aplicando una operación de mutación. Los individuos pasaron a tener la información de los motores de cada tipo condensada en dos genes, uno para la amplitud y el otro para el desplazamiento y una fase para cada motor, de forma que las patas hacían el mismo movimiento pero de forma desfasada. Finalmente mencionar que se volvió a usar selección por torneo.

En la tabla 6 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	550
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	24 Genes
Mutacion	Prob. Individuo	15 %
	Prob. Gen	5 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	3

Tabla 6: Datos del experimento uno de dos controladores

Partiendo de esta base los resultados obtenidos fueron:

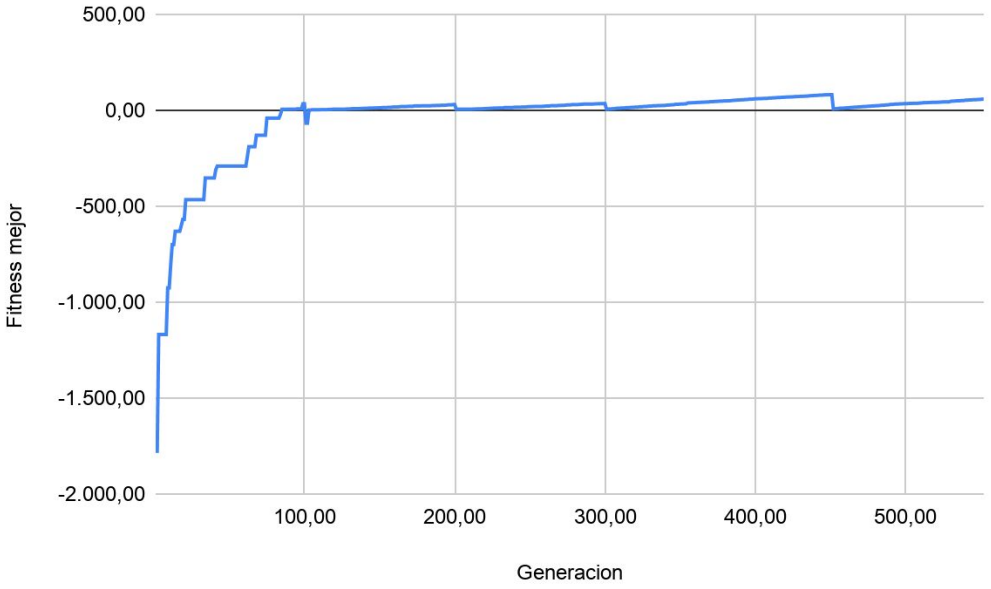


Fig. 34: Fitness del primer conjunto de experimentos con dos controladores.

Como se puede ver el comportamiento es mayormente correcto pero con ciertas irregularidades. Se puede observar cómo en el final de cada experimento y el principio del siguiente hay un escalón, esto se debe a que el propio simulador iba dando valores cada vez mayores según avanzaba la prueba, aunque el individuo se comporta exactamente igual.

Si analizamos el comportamiento del robot al terminar esta ejecución, avanza con el cuerpo muy bajo, las piernas estiradas hacia los lados y dando pequeños saltos. Aún así este no separa todas las patas del suelo, solo las delanteras ligeramente.

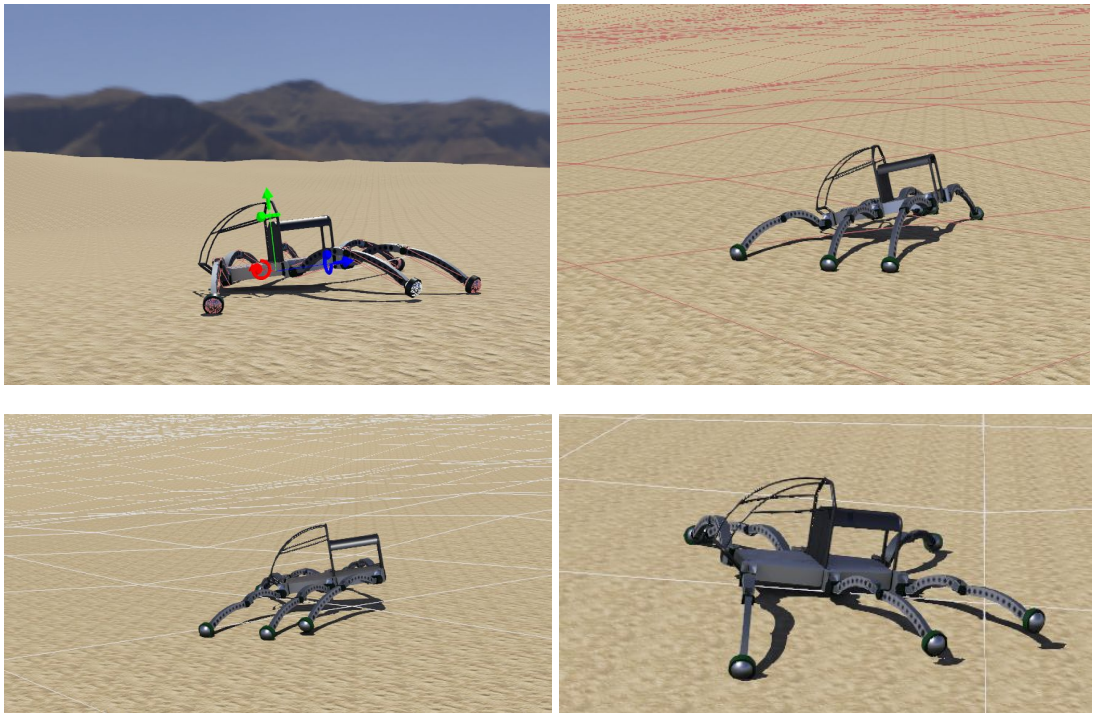


Fig. 35: Método de desplazamiento tras el primer experimento de dos controladores.

En la figura 35 se puede ver la progresión del movimiento del robot.

2. En el segundo grupo de experimentos se parte del resultado del anterior, ya que se considera un buen primer paso desde el que evolucionar. Este tiene dimensiones menores: está compuesto únicamente por dos experimentos, cada uno de 100 generaciones, 50 individuos y las probabilidades de mutación y cruce se mantienen iguales.

   Lo que separa este grupo de experimentos de los anteriores es un cambio radical en la función de fitness. Hasta ahora el desplazamiento se había estado midiendo en horizontal, pero desde este grupo y en adelante se empezó a tomar en cuenta la dirección. Para ello, se hace una proyección de la trayectoria que tome sobre el vector que representa la trayectoria que tomaría si se desplazara en línea recta, después se resta un porcentaje de la distancia que se ha desviado con respecto a

este vector y la puntuación es igual a la función descrita previamente pero cambiando DistRecorrida por esta función de puntuación. De esta forma se puede penalizar que se desvíe de una trayectoria recta. En la figura 36 se puede ver un esquema de esto mismo.

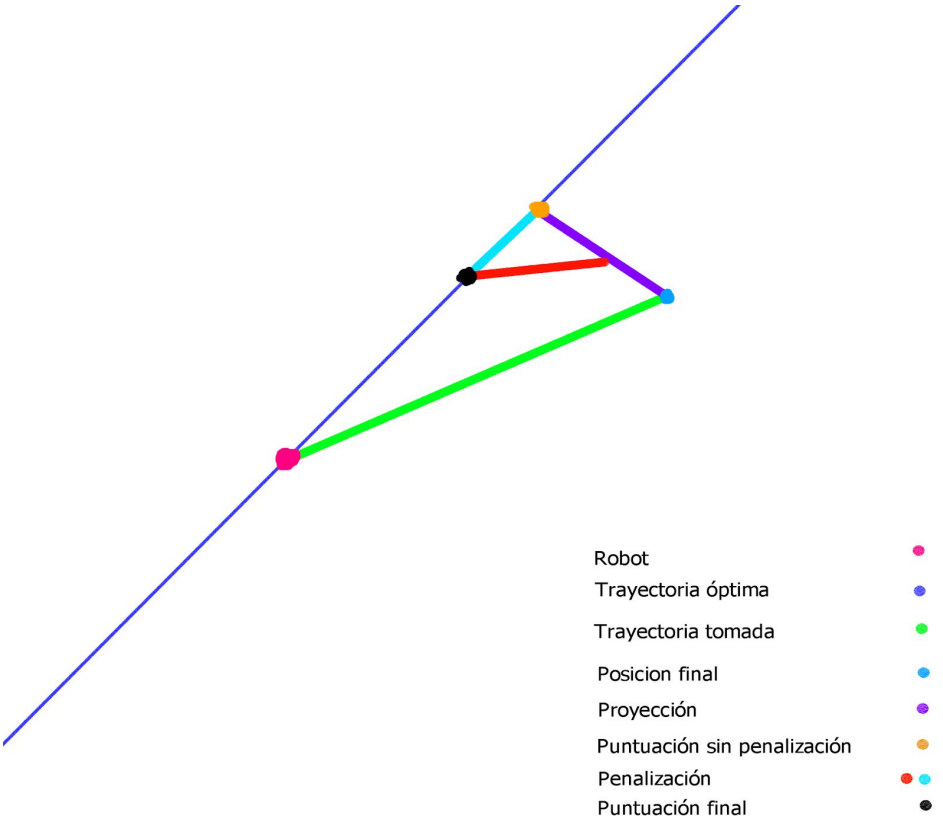


Fig. 36: Esquema de la función de puntuación

En la tabla 7 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	200
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	24 Genes
Mutacion	Prob. Individuo	15 %
	Prob. Gen	5 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	3

Tabla 7: Datos del experimento dos de dos controladores

Teniendo en cuenta esto, los nuevos resultados han sido:

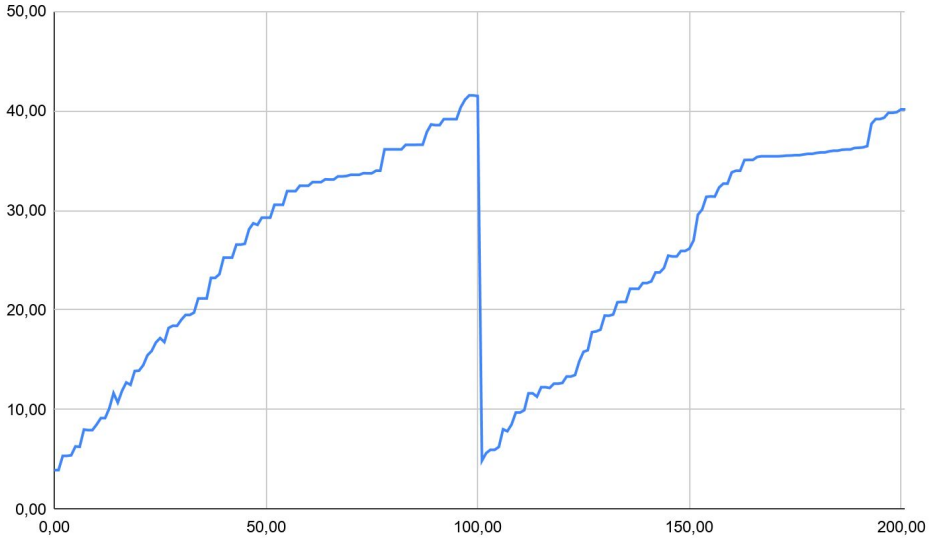


Fig. 37: Fitness del segundo conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

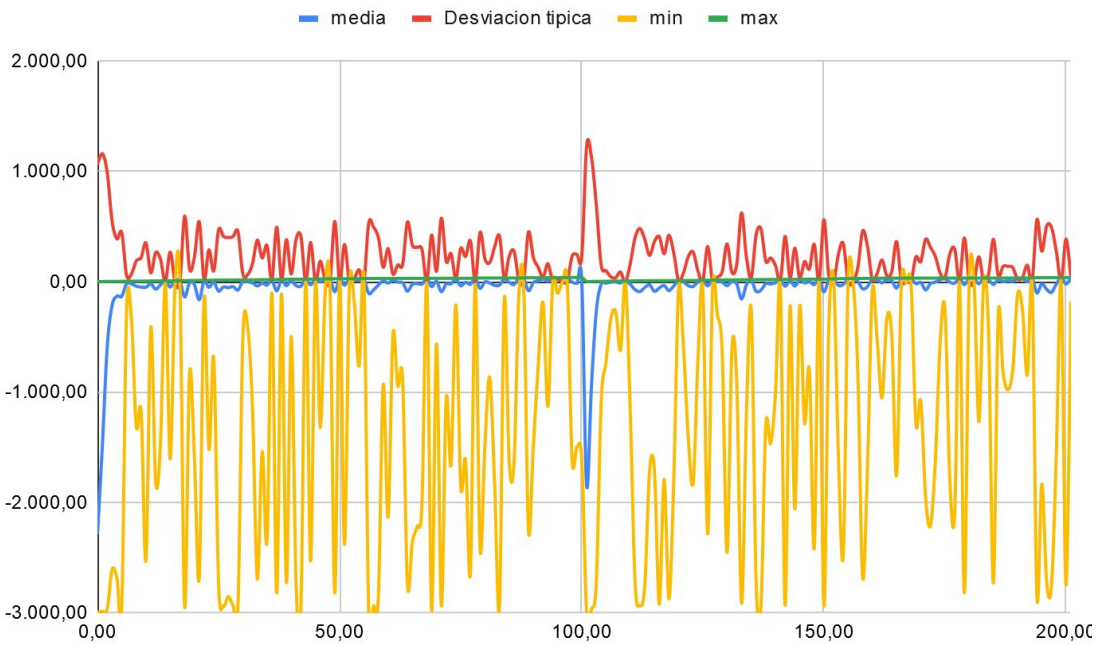


Fig. 38: Fitness de la población del segundo conjunto de experimentos con dos controladores.

Como se puede observar, el ruido introducido por el controlador y la población nueva introducida al principio del experimento son los dos datos más destacables de las tablas. Es importante mencionar que se ha optado por un método nuevo para aplicar el algoritmo genético, el cual permite obtener mejores datos sobre la población y añadir elitismo utilizando funciones y clases de la propia biblioteca.

En cuanto al comportamiento del mejor individuo, podemos ver que al estar basado en el anterior es similar a este, pero ahora en vez de dar saltos hacia delante y caer golpeando la parte delantera del robot, avanza la pata delantera izquierda para, durante el salto, caer con ella y utilizarla de punto de apoyo.

3. El tercer conjunto de experimentos está conformado por siete experimentos de 50 individuos y 100 generaciones como anteriormente y las probabilidades de mutación permanecen constantes.

   En cuanto a las diferencias de este experimento con el anterior, son principalmente dos, uno de los cambios se encuentra en el *setup* del experimento ya que antes el robot iniciaba desde una posición de *standby* para realizar el desplazamiento, y ahora el robot queda en la posición en la que debe empezar a andar durante 5 segundos para asegurar que parta de una postura estable antes de empezar a andar. El otro cambio reside en la función de fitness, ya no se suman los valores de los sensores en cada instante de tiempo sino -1 por cada vez que haya menos de 3 sensores apoyados, que matemáticamente se representaría como:

$$suma = \sum_{i=0}^{1000} (round(\sum_{j=0}^{6} (sensor_{ij})/6) - 1)$$

   Siendo *i* los instantes de tiempo y *j* los sensores.

   En la tabla 8 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	700
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	24 Genes
Mutacion	Prob. Individuo	15 %
	Prob. Gen	5 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	3

Tabla 8: Datos del experimento tres de dos controladores

Habiendo realizado estos cambios, los resultados son los siguientes:

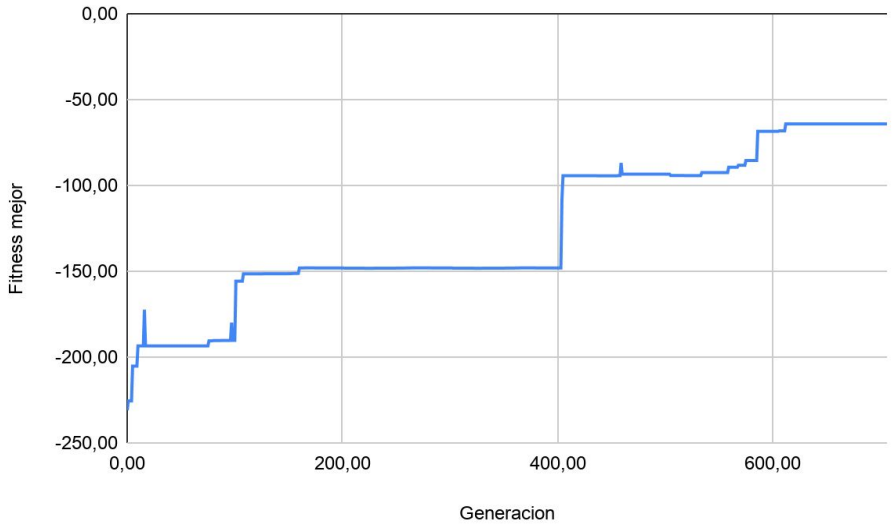


Fig. 39: Fitness del tercer conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

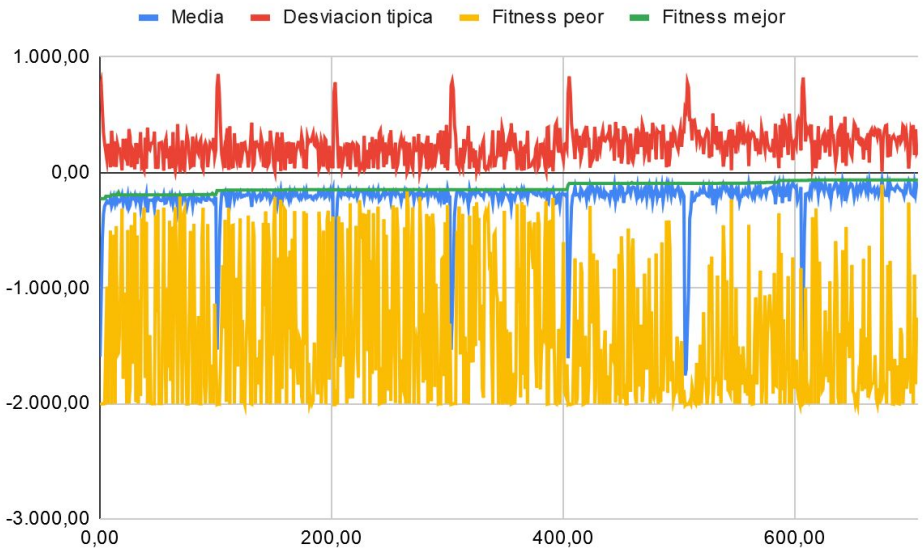


Fig. 40: Fitness de la población del tercer conjunto de experimentos con dos controladores.

Como se puede observar el ruido que variaba los resultados se ha reducido considerablemente, únicamente apareciendo en forma de algunos picos, los valores vuelven a ser negativos debido a que al andar con las patas extendidas había momentos en los que los sensores no terminaban de recibir el impacto con el suelo, como en la figura 41.

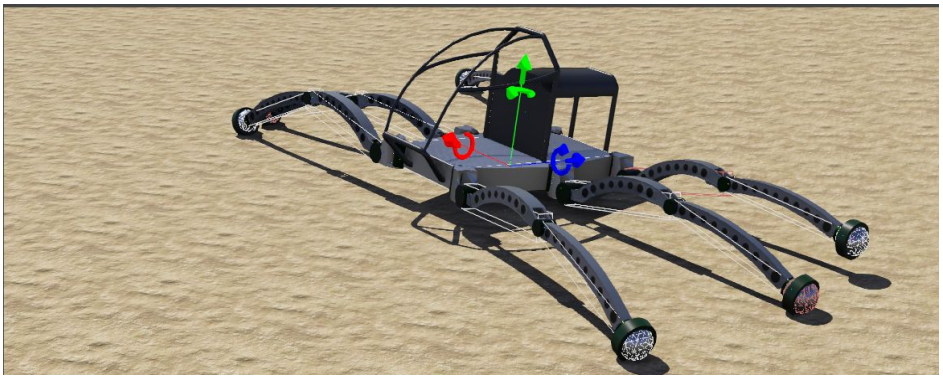


Fig. 41: Posición del individuo del tercer grupo de experimentos.

En cuanto al comportamiento del robot, es similar a los anteriores, pero en vez de dar saltos tan marcados como anteriormente, empieza a intercalar el movimiento de las patas, siendo cada vez un sistema de marcha más estable pero con la disposición de las patas aún demasiado extendida.

4. El cuarto conjunto de experimentos vuelve a ser uno corto, teniendo solo dos experimentos, con las mismas dimensiones que en el caso anterior.

   Las variaciones de este experimento con el anterior radican en añadir dos acciones ilegales que penalizan en la función de fitness, estas son tener las rodillas demasiado extendidas y que tengan poca variación en su movimiento (la amplitud y el desplazamiento de la función seno que rige el movimiento de las mismas).

En la tabla 9 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	200
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	24 Genes
Mutacion	Prob. Individuo	25 %
	Prob. Gen	20 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	3

Tabla 9: Datos del experimento cuatro de dos controladores

Con esto, los resultados son:

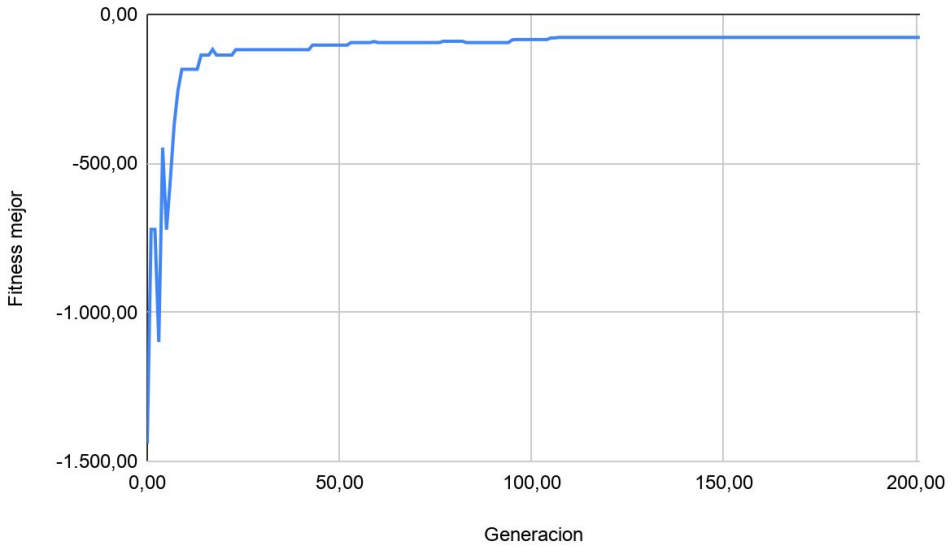


Fig. 42: Fitness del cuarto conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

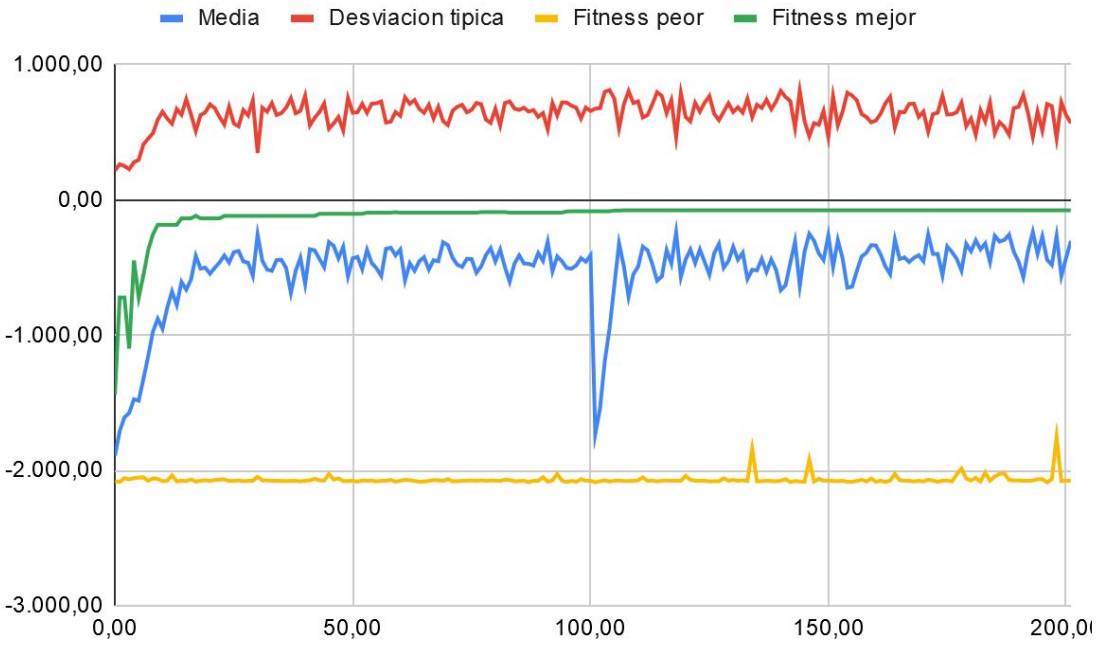


Fig. 43: Fitness de la población del cuarto conjunto de experimentos con dos controladores.

Las gráficas vuelven a mostrar valores malos por la nueva penalización pero pronto los individuos vuelven a aprender y a mejorar su fitness, ya que simplemente se deben variar un poco dos de los parámetros para evitar las penalizaciones indicadas.

El comportamiento del individuo resultante es similar pero algo menos estable, ya que no ha doblado demasiado las rodillas pero si que las mueve llevando a que los saltos anteriormente mencionados sean mayores y más frecuentes y a que en determinadas partes del terreno se pueda volcar, cosa que al evaluarse

solo 10 segundos de ejecución no se controla, pero que posteriormente se solucionará.

5. En el quinto experimento se parte desde cero, empezando con individuos inicializados de forma aleatoria y solo se ejecutan 100 generaciones y 50 individuos con las mismas probabilidades.

   En este caso se añaden nuevamente acciones ilegales, las cuales son mover el eje horizontal de los hombros a valores superiores a $\frac{\pi}{2}$ e inferiores a $-\frac{\pi}{2}$ radianes, ya que de hacerlo colisionaría consigo mismo. Además se modificaron las penalizaciones del experimento anterior, las cuales eran valores constantes y se pusieron todas proporcionales a cuánto estaban haciéndolo mal para que fuera más claro para el algoritmo.

   En la tabla 10 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	100
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	25 Genes
Mutacion	Prob. Individuo	35 %
	Prob. Gen	50 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	3

Tabla 10: Datos del experimento cinco de dos controladores

Con las nuevas restricciones los resultados fueron:

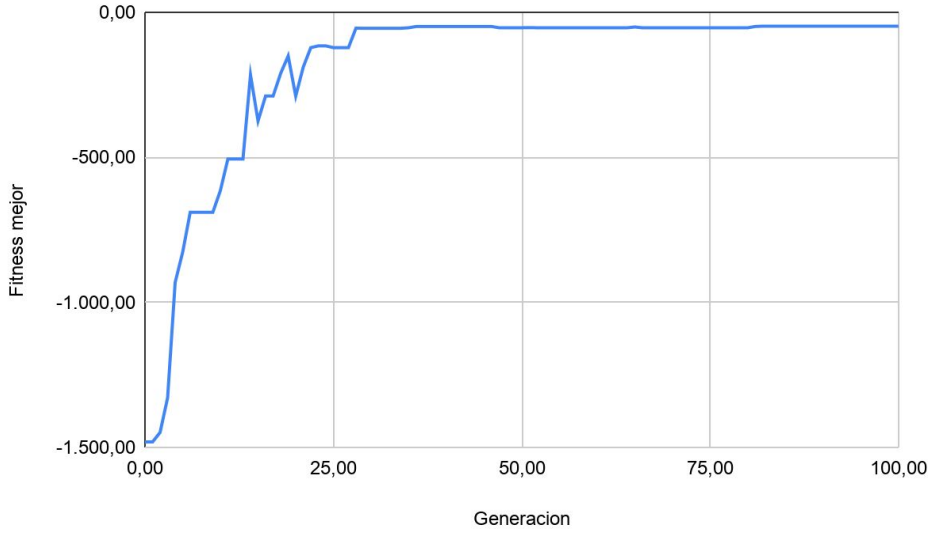


Fig. 44: Fitness del quinto conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

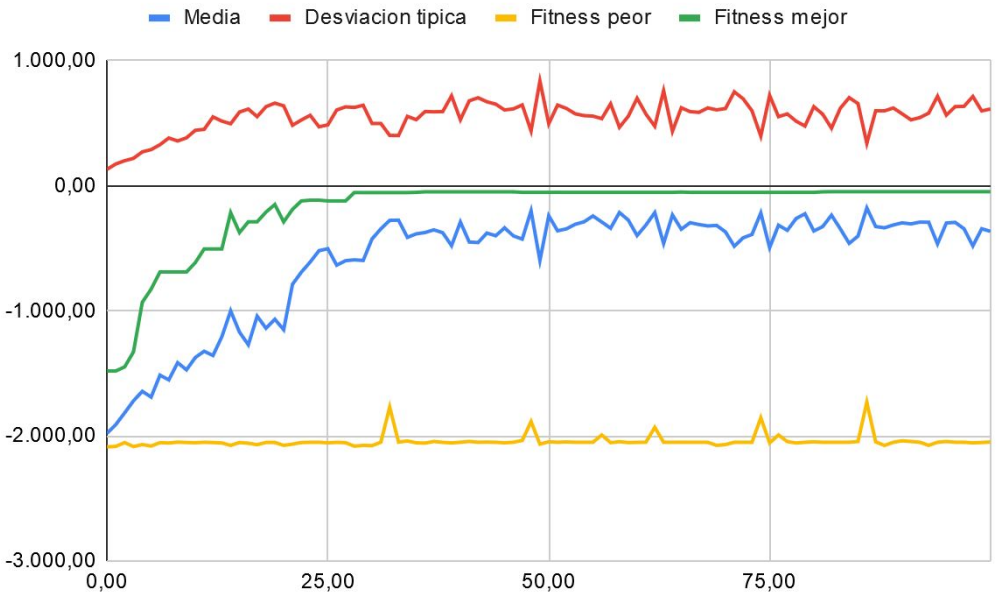


Fig. 45: Fitness de la población del quinto conjunto de experimentos con dos controladores.

Como se puede ver, gracias a la nueva función de fitness los individuos mejoran muy rápido y la media del fitness también es razonablemente alta.

Por desgracia, si bien el individuo resultante es estable es incapaz de andar, apenas levanta las patas, las tiene muy extendidas y cruzadas la mayoría del tiempo como se puede ver en la figura 45.

Fig. 46: Postura el mejor individuo del quinto grupo de experimentos.

Aun con esto, la rápida evolución indica que la función de fitness con las acciones ilegales es una buena base de la cual partir para los siguientes experimentos.

6. Este grupo de experimentos vuelve a partir de la función optimizada anteriormente pero varían las probabilidades de mutación, aumentando a un 35% de mutar cada individuo y a un 50% de mutar cada gen, haciendo que explore aún más soluciones. Los operadores no han sufrido ningún cambio.

En la tabla 11 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	1200
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	25 Genes
Mutacion	Prob. Individuo	35 % - 30%
	Prob. Gen	10 % - 5%
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	3

Tabla 11: Datos del experimento seis de dos controladores

Por lo tanto los resultados obtenidos son los siguientes:

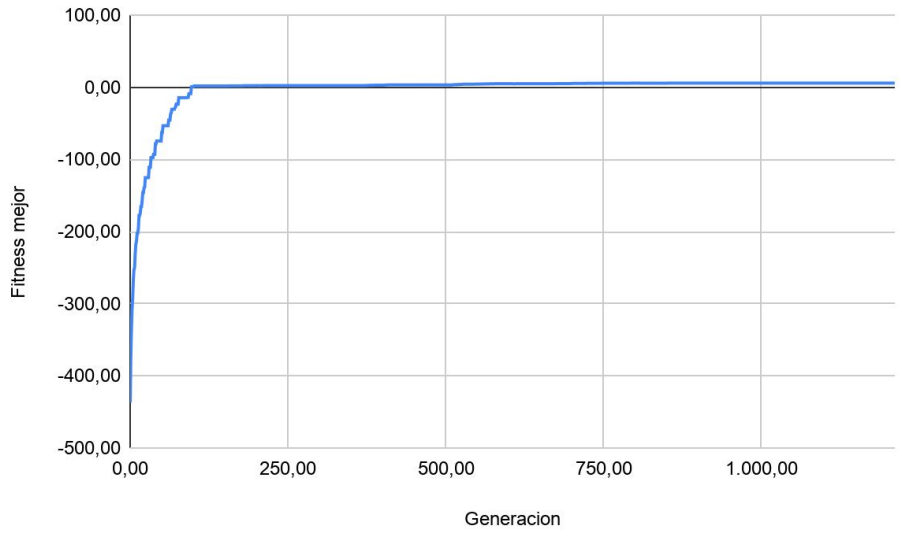


Fig. 47: Fitness del sexto conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

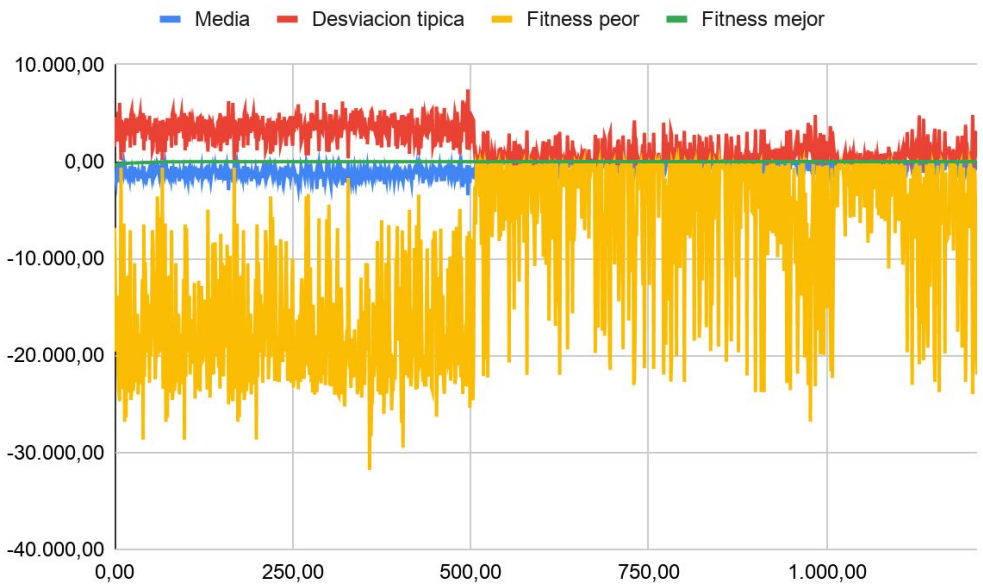


Fig. 48: Fitness de la población del sexto conjunto de experimentos con dos controladores.

Como se puede ver en este caso la evolución es razonablemente buena, empezando por valores cercanos a -400 y llegando a valores positivos de alrededor de 6. Es muy destacable la variación en los valores mínimos y la desviación típica a partir del experimento número 6 (iteración 500), esto sucede ya que las probabilidades de mutación se redujeron a 30% para los individuos y 5% para cada gen, esto se hizo con el objetivo de que cada individuo mutase menos, ya que estaba teniendo problemas para evolucionar llegado a ese punto.

En cuanto al comportamiento del individuo, a partir del primer experimento se puede observar que consigue una puntuación positiva con valores alrededor de 0,67 y analizando al individuo se puede ver que es el primero realmente estable como se puede observar en la figura 49.

Fig 49: Postura del mejor individuo del quinto grupo de experimentos.

Esta postura se va a mantener a lo largo de las generaciones, variando en el movimiento de las patas, y la fase de cada articulación. Al final de la evolución el mejor individuo terminó desviando su trayectoria hacia la izquierda y fue razonablemente lento. Esto llevó a necesitar empezar la evolución desde cero para solventar problemas de los cuales no era capaz de salir. Además, con el siguiente reinicio se resolvería un problema estructural el cual era fácil de solucionar pero no se había visto necesario hasta ahora, esto es que el sentido del vector sobre el que debe de marchar el robot estaba invertido.

7. Como se mencionó previamente, el grupo de experimentos número siete vuelve hacia atrás, en este caso partiendo del resultado del individuo final del experimento 19 (el primer individuo estable obtenido en el grupo de experimentos anterior).

   Partiendo de esto se realizaron dos cambios en las dimensiones de los experimentos, para empezar se redujo la población a la mitad, esto es debido a

que el tiempo que se evalúa aumentó de 10 a 60 segundos, con esto se esperaba evaluar si la marcha que realizara el robot fuera recta o al menos detectar si a lo largo de más tiempo se desviaba. Este grupo se extiende a lo largo de 400 generaciones.

Por otro lado, en la función de fitness, se cambió el sentido del vector que representa la dirección en la que se podía mover de forma óptima el robot.

En la tabla 12 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	400
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	25 Genes
Mutacion	Prob. Individuo	35 %
	Prob. Gen	10 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	20 %
Selección	Tamaño torneo	3

Tabla 12: Datos del experimento siete de dos controladores

Con estos cambios los resultados han sido:

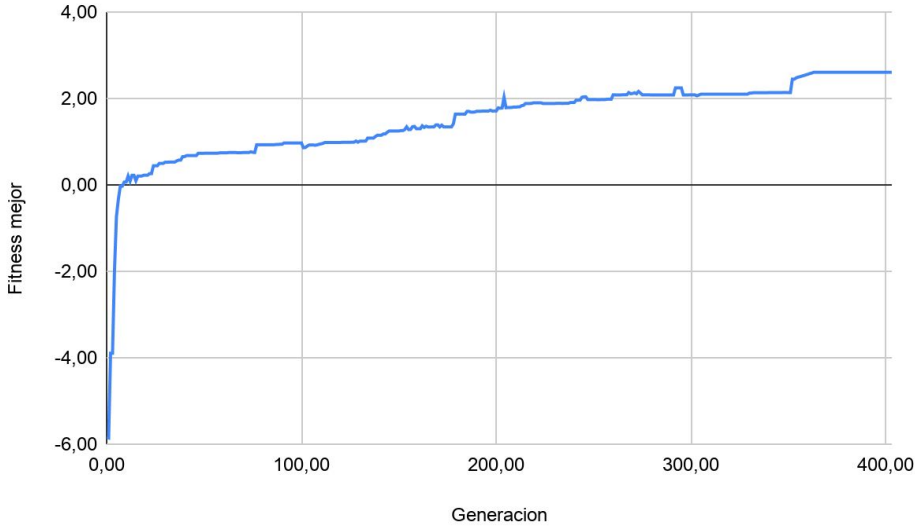


Fig. 50: Fitness del séptimo conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

Fig. 51: Fitness de la población del séptimo conjunto de experimentos con dos controladores.

Como se puede ver la evolución fue bastante buena, llegando a resultados positivos. El fitness del mejor individuo empezó en un valor alto, -6 y llega a 2, esto sucede porque el individuo del que se partió apenas se desplazaba en dirección contraria pero es bastante estable como se puede ver en la figura 52.

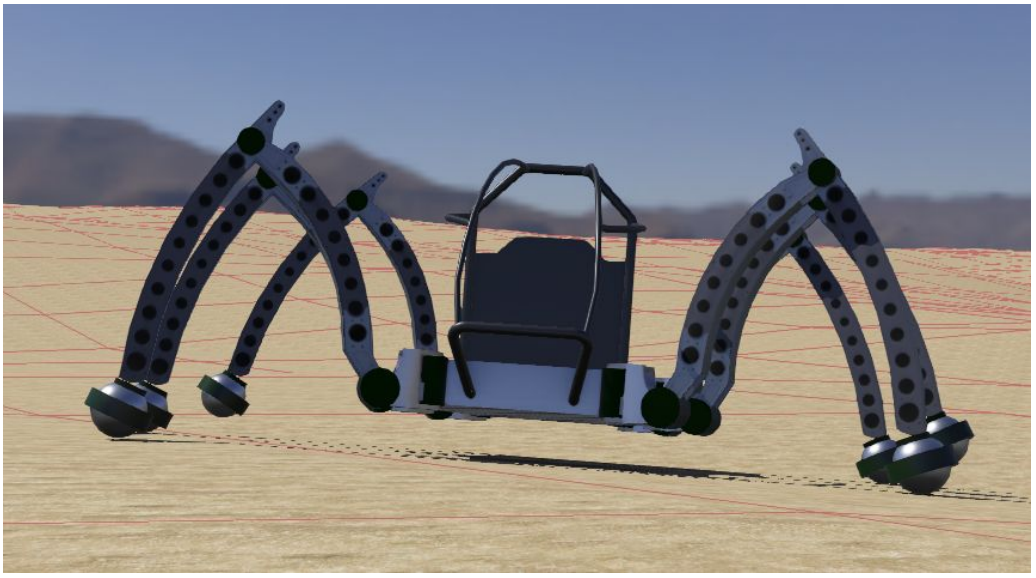


Fig. 52: Postura el mejor individuo del séptimo grupo de experimentos.

En cuanto al comportamiento del robot, avanza de una forma muy similar al mejor individuo del grupo anterior, el único defecto que lleva a dejar el experimento en este punto es que el cuerpo del robot está demasiado bajo como para adaptarse a todo tipo de curvaturas en el terreno.

8. Este grupo de experimentos vuelve a empezar en el resultado del experimento 19, manteniendo las dimensiones definidas anteriormente. Este grupo se extiende a lo largo de 700 generaciones.

   En este grupo de experimentos se añadieron nuevas restricciones sobre qué acciones son ilegales, considerando límites en la apertura de todos los motores.

En la tabla 13 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	700
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	25 Genes
Mutacion	Prob. Individuo	35 %
	Prob. Gen	10 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	30 %
Selección	Tamaño torneo	3

Tabla 13: Datos del experimento ocho de dos controladores

Con estas modificaciones, se consiguen los siguientes resultados.

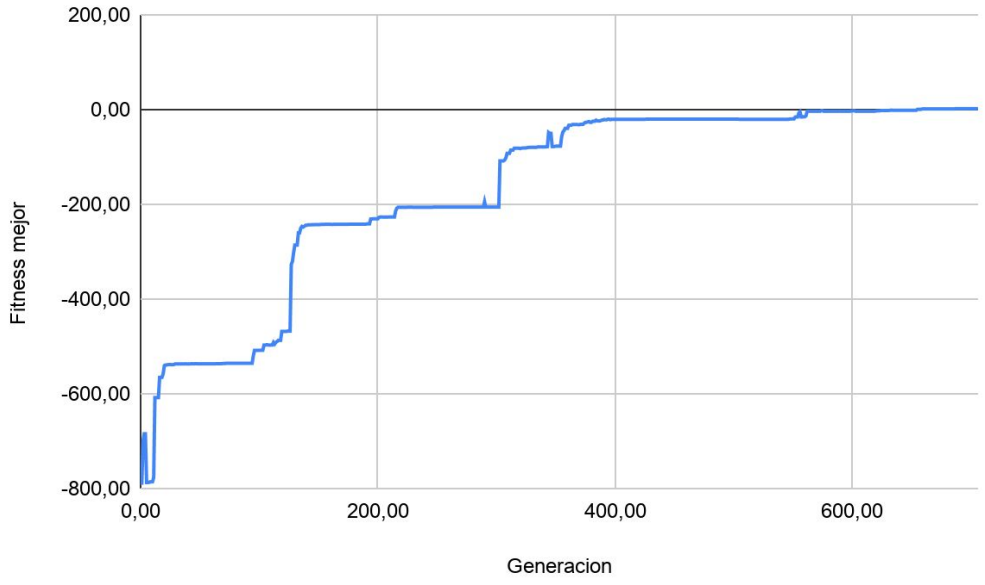


Fig. 53: Fitness del octavo conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

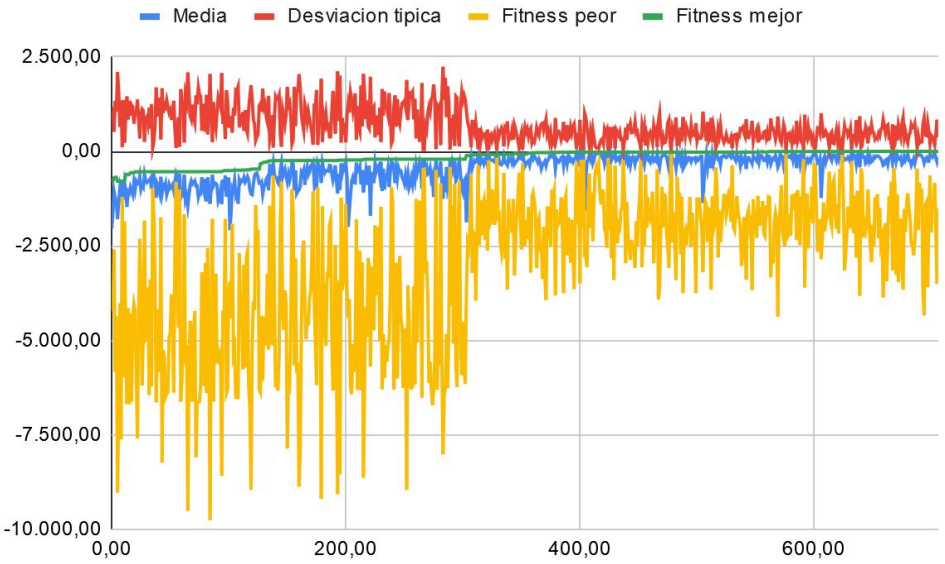


Fig. 54: Fitness de la población del octavo conjunto de experimentos con dos controladores.

Como se puede ver la evolución es similar a la anterior, llegado a valores similares, pero desde un valor inicial mucho menor.

Donde sí se puede hallar una mejora es en su comportamiento, pues si bien aún no es capaz de coordinar bien el movimiento de sus patas, la postura es mucho mejor como se puede observar en la figura 55.

Fig. 55: Postura el mejor individuo del octavo grupo de experimentos.

9. El siguiente grupo de experimentos parte directamente del anterior, constando de 9 experimentos de 100 generaciones con las mismas dimensiones que los anteriores.

   Lo que separa este grupo del anterior es la función de fitness a la cual se le han añadido nuevos parámetros ilegales, en este caso se ha puesto que el motor que desplaza el hombro realizando movimiento vertical tiene que tener un mínimo de variación, cosa que de no ser así impide un correcto desplazamiento en el mejor individuo de la generación anterior.

En la tabla 14 se puede encontrar un resumen de los datos del experimento.

Dimensiones	Generaciones	1000
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	25 Genes
Mutacion	Prob. Individuo	35 %
	Prob. Gen	10 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	30 %
Selección	Tamaño torneo	3

Tabla 14: Datos del experimento nueve de dos controladores

Con estos cambios el resultado es el que sigue:

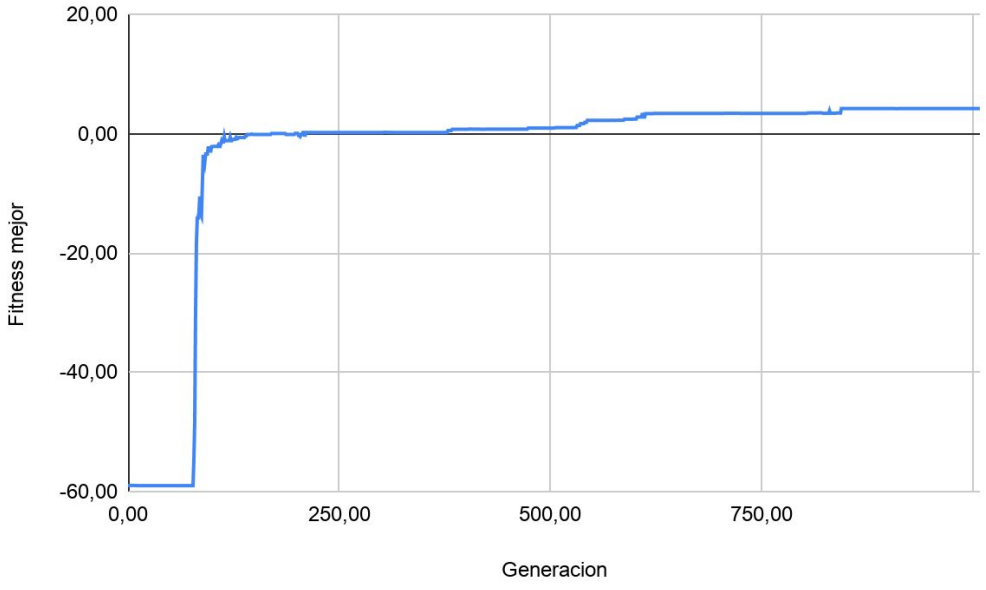


Fig. 56: Fitness del noveno conjunto de experimentos con dos controladores.

Y el comportamiento de la población:

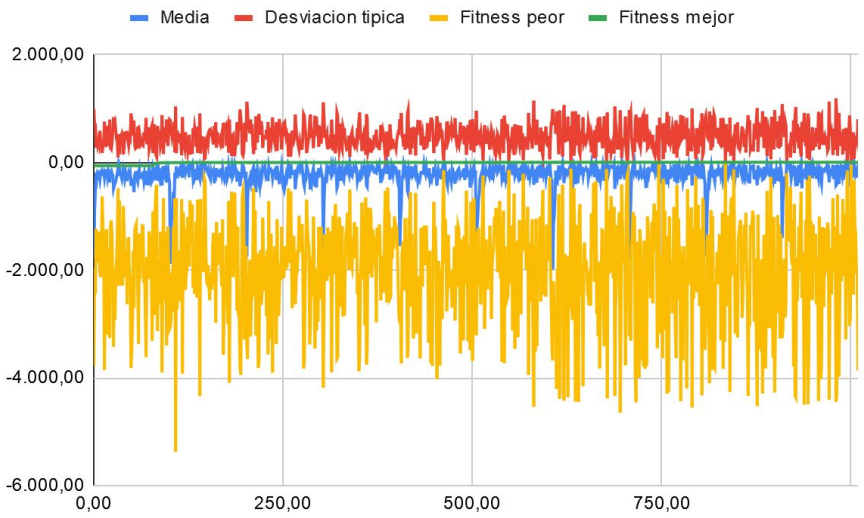


Fig. 57: Fitness de la población del noveno conjunto de experimentos con dos controladores.

Como se puede observar en las gráficas, las primeras generaciones se atascaron en el valor -58, esto se debe a la nueva restricción que se indicó. A partir del valor previamente mencionado los valores obtenidos fueron positivos. El mayor valor alcanzado fue 4.22 pero se necesitaron alrededor de 850 generaciones para hacerlo desde el valor 0.067 que es el primer valor positivo que se obtuvo. Además mencionar que el valor 4.22 se mantuvo a lo largo de varios cientos de generaciones más que no se añadieron a la tabla debido a que no supusieron información adicional.

En cuanto al comportamiento del robot, este realizaba un movimiento rítmico en el que desplazaba el cuerpo hacia delante y atrás y movía las patas de la parte izquierda como si intentase andar, pero las de la parte derecha realizaban un movimiento básico, todas a la vez, impidiendo que andase. Esto último unido al

estancamiento son las razones de terminar el experimento llegado a este punto y empezar otro desde un valor anterior, con la esperanza de que tomase una ruta diferente.

4.5 Modelo final

Debido a los resultados anteriores, se puede observar que el mayor problema de los modelos anteriores que llevan a que el individuo no pueda caminar bien es que no es capaz de optimizar la fase de ambos lados del robot.

Para poder solventar esto se han probado varias soluciones intermedias, entre ellas:

- En un principio se trató de solventar el problema cambiando las probabilidades de mutación de los genes de los individuos mutados, con la esperanza de que si se subía o bajaba la probabilidad de ello se pudieran optimizar las fases de los motores necesarias sin alterar el resto de genes que ya estaban optimizados, pero dado que esto depende de la probabilidad, no se consiguieron resultados demasiado buenos.

- Otra de las posibles soluciones que se probaron fue partir de experimentos anteriores, esperando que con las modificaciones en la función de fitness se lograra un resultado mejor, pero esto solo llevó a comportamientos raros del robot como encoger las patas hacia dentro. Esto en vez de solventar los problemas que había añadió más, así que se descartó esta solución.

- La siguiente solución por la que se optó fue reducir la cantidad de fases de las patas a 6, uno para cada motor de 3 patas, de forma que cada pata se ajustara a uno de estos de forma alternada como demuestra la figura 58, pero si bien esto permite al robot avanzar, hacía sus movimientos demasiado toscos, perdiendo bastante estabilidad con ello.

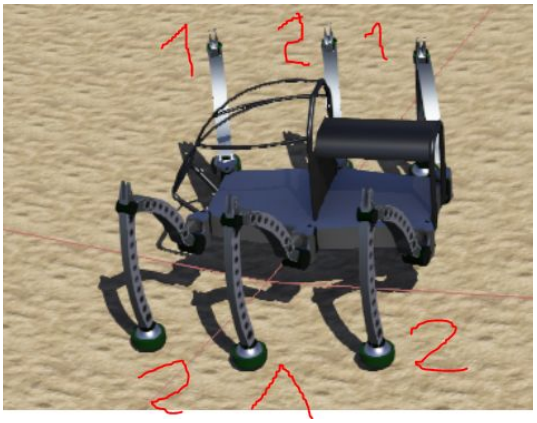


Fig 58: Reparto de las fases en la posible solución.

Como ninguna de las alternativas planteadas anteriormente fueron fructíferas, no se guardaron los registros de ellas, pero como el resultado que mejor se acercaba a resolver el problema era el de reducir el número de fases, para el resultado final se optó por seguir con la idea. Ya que con solo las fases de dos patas era bastante inestable se optó por hacerlo de tres. Para esto se ajustó el controlador de forma que el comportamiento del lado derecho fuera igual que el del lado izquierdo pero desfasado π radianes, de forma que los movimientos de los motores fueran complementarios. Con esto, cada par de patas tenía un comportamiento diferente permitiendo que se mantuviera estable, que se redujera la cantidad de parámetros a optimizar, y que el desplazamiento fuera más recto que en los anteriores experimentos porque los movimientos eran complementarios. Adicionalmente se cambió la probabilidad de la mutación de cada uno de los genes.

Los datos específicos del experimento se pueden ver en la tabla 15.

Dimensiones	Generaciones	300
	Tamaño población	50 Individuos
Individuo	Tipo de gen	Valor [0,1]
	Tamaño	25 Genes
Mutacion	Prob. Individuo	35 %
	Prob. Gen	6.6 %
Cruce	Prob. Individuo	60 %
	Prob. Gen	30 %
Selección	Tamaño torneo	3

Tabla 15: Datos de experimentación del modelo final.

Habiendo realizado estos cambios, los resultados han sido:

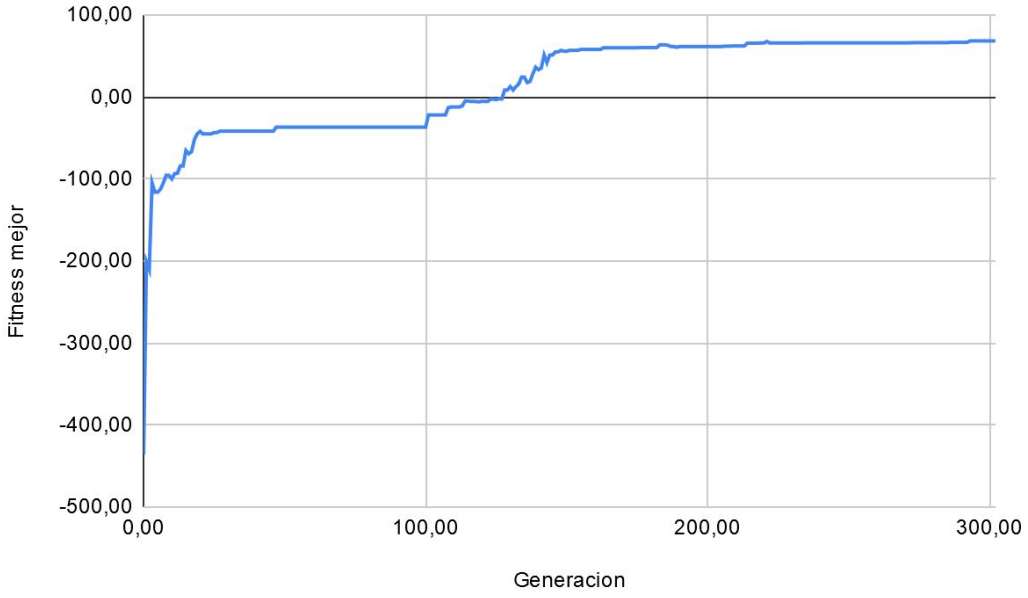


Fig. 59: Fitness del modelo final.

Y el comportamiento de la población:

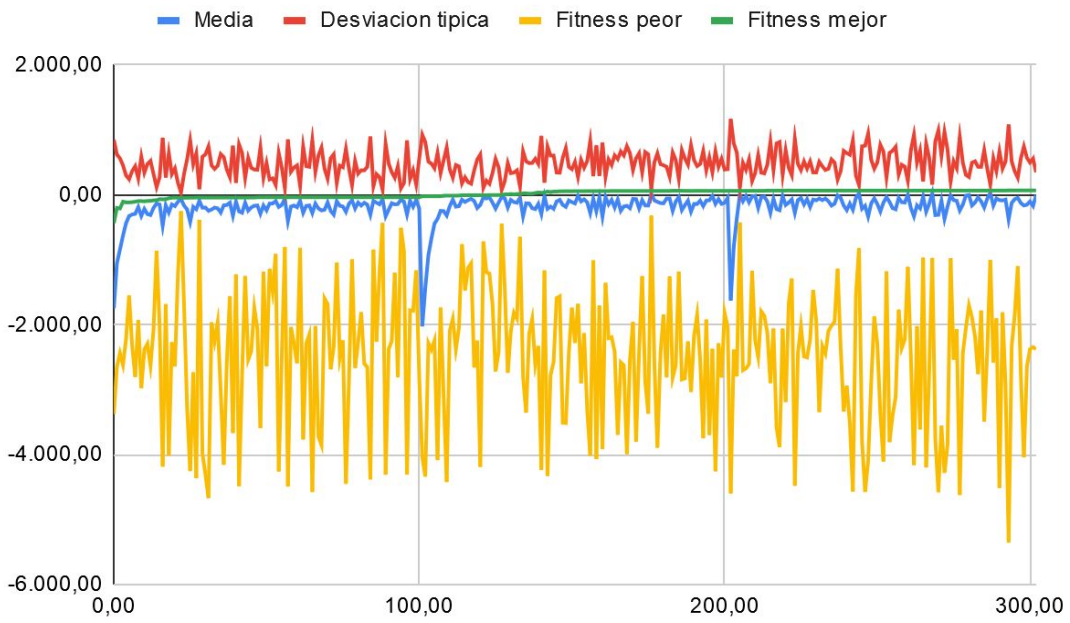


Fig. 60: Fitness de la población del modelo final.

Como se puede ver en las gráficas el modelo final termina con un valor de fitness de 68.5, esto nos deja ver que es un buen individuo ya que la función de fitness indica que camina una gran distancia y en una línea más o menos recta y no comente ninguna de las acciones ilegales, las cuales pueden ser poder doblar las articulaciones de forma que colapse en sí mismo o de alguna forma que no pueda mantenerse estable.

Una vez se analiza al individuo se puede ver que la predicción hecha en base al resultado del fitness del individuo es correcta, las dos patas delanteras y las dos centrales sirven para realizar un proceso de marcha en el cual se alternan las patas diagonales para andar y las traseras empujan al robot hacia delante, rotando el cuerpo del robot según avanza en el eje Z. No obstante, debido a la forma en la que se mueven las patas delanteras, esta rotación no tiene repercusión negativa en la estabilidad del

robot ni en el sentido de la marcha, indicando que el objetivo general planteado al comienzo del trabajo se ha cumplido.

Capítulo 5: Entorno económico

En este capítulo se hablará sobre la planificación del trabajo, cómo este se ha distribuido y el presupuesto que está asociado al proyecto.

5.1 Planificación

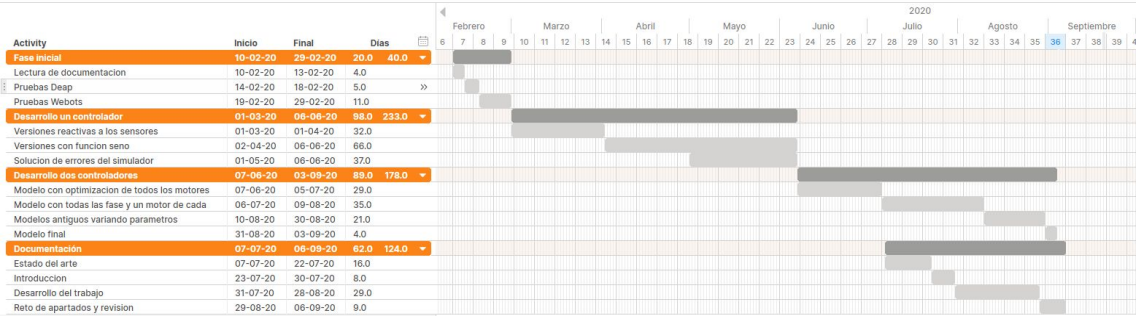
Cuando se habla de planificación lo primero que debe hacerse es definir las fases del desarrollo del proyecto, de esta forma se concluyen las siguientes:

- **Fase inicial.** Esta fase contempla la búsqueda inicial de la información y las primeras pruebas con las librerías utilizadas y con el simulador. Esta fue la primera en llevarse a cabo.

- **Desarrollo un controlador.** Esta fase ha sido la primera en llevarse a cabo cuyos resultados se contemplan dentro del proyecto, contempla las primeras soluciones y modelos creados con un único controlador.

- **Desarrollo dos controladores.** Esta fase se ha llevado a cabo después de la fase del desarrollo de un controlador, contempla desde las primeras soluciones con dos controladores hasta el modelo final.

- **Documentación.** Esta fase se ha ido llevando a cabo conjuntamente con la fase de desarrollo de dos controladores y cubre la realización de este documento.



Activity	Inicio	Final	Días	
Fase inicial	10-02-20	29-02-20	20.0	40.0
Lectura de documentacion	10-02-20	13-02-20	4.0	
Pruebas Deap	14-02-20	18-02-20	5.0	
Pruebas Webots	19-02-20	29-02-20	11.0	
Desarrollo un controlador	01-03-20	06-06-20	98.0	233.0
Versiones reactivas a los sensores	01-03-20	01-04-20	32.0	
Versiones con funcion seno	02-04-20	06-06-20	66.0	
Solucion de errores del simulador	01-05-20	06-06-20	37.0	
Desarrollo dos controladores	07-06-20	03-09-20	89.0	178.0
Modelo con optimizacion de todos los motores	07-06-20	05-07-20	29.0	
Modelo con todas las fase y un motor de cada	06-07-20	09-08-20	35.0	
Modelos antiguos variando parametros	10-08-20	30-08-20	21.0	
Modelo final	31-08-20	03-09-20	4.0	
Documentación	07-07-20	06-09-20	62.0	124.0
Estado del arte	07-07-20	22-07-20	16.0	
Introduccion	23-07-20	30-07-20	8.0	
Desarrollo del trabajo	31-07-20	28-08-20	29.0	
Reto de apartados y revision	29-08-20	06-09-20	9.0	

Fig. 61: Diagrama de Gantt del proyecto

La figura 61 representa el diagrama de Gantt del proyecto donde se puede ver cómo se han distribuido las diferentes fases del proyecto definidas anteriormente.

5.2 Presupuesto

Electricidad			
Potencia contratada			63.05 €
Electricidad			17.36391572 €
Equipo eléctrico			4.50 €
Impuesto electricidad			0.2304653204 €
IVA			17.88 €
Total:			103.02 €
Equipo			
Ordenador			1528€
Pantallas			440€
Teclado			70€
Ratón			50€
Internet			150€
Total			2238€
Salario			
Trabajadores		Sueldo / h	40.87€
Total:			20436.63€
Total:			27069.52€

Tabla 16: Costes del proyecto.

Capítulo 6: Conclusiones y trabajos futuros

6.1 Conclusiones

Tras la investigación realizada para entender el problema, se puede concluir que la perspectiva que aporta la RoboCup sobre la importancia de la mejora de las tareas de rescate, el uso de robots para las mismas y la existencia de los robots zoomórficos, los cuales a día de hoy se utilizan para investigar métodos de locomoción alternativa, son de gran importancia para el futuro. Si se sigue investigando en ese área en un futuro se podrán realizar rescates en terrenos accidentados como por ejemplo edificios derruidos, los cuales son un gran riesgo porque pueden derrumbarse partes en medio de las operaciones, de forma mucho más segura.

No obstante, para poder llegar a un objetivo como sea aplicar un robot zoomórfico a tareas de rescate primero se debe partir de un problema más básico. En este caso se ha planteado una adaptación a terreno desigual del sistema de marcha de un robot porque es un problema simplificado desde el cual se puede partir para, mediante el uso de más sensores y actuadores, llegar a un robot que pueda desenvolverse bien en una situación de riesgo.

Es importante mencionar que a lo largo del desarrollo del trabajo se ha podido comprobar de primera mano la importancia de los robots con métodos de locomoción diferentes a ruedas, ya que gracias a la estabilidad aportada por las patas se ha podido entrenar en terreno irregular sin riesgo de que se volcase ni que pudiese suceder algún percance similar, ya que permite que el cuerpo del robot esté lo más bajo posible en función de la forma del terreno que deba abordar.

En cuanto a los objetivos del trabajo, se puede ver que a través de la práctica adquirida en el desarrollo del mismo se ha obtenido la soltura necesaria para el desarrollo del

proyecto, cosa que se ha demostrado a la hora de solventar los diferentes problemas que han ido surgiendo a lo largo del desarrollo.

Por otro lado el controlador elegido para resolver el problema, el cual está optimizado, es el resultante de diferentes iteraciones que han ido mejorando para una optimización más rápida y eficiente.
La solución final ha sido resultante del entrenamiento a lo largo de cientos de horas, con varias soluciones que no han resultado y un experimento final el cual se presenta como la solución final del proyecto.

Finalmente, se ha ido haciendo un análisis de los resultados obtenidos según conjuntos de experimentos, hablando del comportamiento de su fitness y del de su individuo final.

Adicionalmente mencionar la complejidad de la adaptación del proceso de marcha del robot una vez realizado este trabajo, y cómo a lo largo del desarrollo se ha ido haciendo notorio que si bien el uso de algoritmos genéticos es una herramienta muy potente para la optimización de comportamientos, es muy necesario definir un modelo correcto que permita que la optimización sea eficiente, ya que al final terminan dependiendo de la probabilidad y esto tiene una influencia notoria en los resultados.

6.2 Trabajos futuros

Si se habla de trabajos futuros es obvio que, si bien el resultado es aceptable, no es óptimo ya que aunque el robot es capaz de andar de forma recta en terreno irregular, aún falta que partiendo del resultado obtenido en este trabajo el robot adapte el comportamiento individual de cada pata a la forma del terreno por la que pasa, de forma que no solo sea capaz de navegar por terreno irregular, sino que también sea capaz de pasar por irregularidades más notables del terreno como puedan ser rocas y otros elementos.

Por otro lado, como se ha mencionado previamente, este trabajo estaba enfocado desde el punto de vista de poder iterar más sobre este robot y terminar adaptándolo a tareas de rescate, por lo tanto también sería interesante añadir algún tipo de actuador, como por ejemplo un brazo robótico que le permita interactuar con elementos de un desastre, como escombros y rocas.

Por último, siguiendo la tónica del rescate mencionada anteriormente, sería interesante añadir cámaras y algún tipo de sistema de visión artificial que permita localizar a personas que puedan estar atrapadas por derrumbamientos u otros desastres.

Bibliografía

[1] "Lie Yukou", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Lie_Yukou. [Accessed: 06- Jul- 2020].

[2] "Robótica", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Rob%C3%B3tica. [Accessed: 03- Jul- 2020].

[3] "Herón de Alejandría", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Her%C3%B3n_de_Alejandr%C3%ADa. [Accessed: 03- Jul- 2020].

[4] 2020. [Online]. Available: https://www.youtube.com/watch?v=-wR6jAYgVPM. [Accessed: 03- Jul- 2020].

[5] "10 momentos para recordar la historia de la robótica - Como Funciona Que", *Como Funciona Que*, 2020. [Online]. Available: https://comofuncionaque.com/historia-de-la-robotica/. [Accessed: 03- Jul- 2020].

[6] "Al Jazarí", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Al_Jazar%C3%AD. [Accessed: 03- Jul- 2020].

[7] "Leonardo da Vinci", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Leonardo_da_Vinci. [Accessed: 03- Jul- 2020].

[8] "Robot de Leonardo", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Robot_de_Leonardo. [Accessed: 03- Jul- 2020].

[9] "Tanaka Hisashige", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Tanaka_Hisashige. [Accessed: 03- Jul- 2020].

[10] "Elektro", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Elektro. [Accessed: 03- Jul- 2020].

[11] "Elmer and Elsie (robots)", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Elmer_and_Elsie_(robots). [Accessed: 03- Jul- 2020].

[12] "Unimate", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Unimate. [Accessed: 03- Jul- 2020].

[13] "Nao (robot)", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Nao_(robot). [Accessed: 03- Jul- 2020].

[14] "Sophia (robot)", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Sophia_(robot). [Accessed: 03- Jul- 2020].

[15] G. M. Nelson, R. D. Quinn, R. J. Bachmann, W. C. Flannigan, R. E. Ritzmann and J. T. Watson, "Design and simulation of a cockroach-like hexapod robot," Proceedings of International Conference on Robotics and Automation, Albuquerque, NM, USA, 1997, pp. 1106-1111 vol.2, doi: 10.1109/ROBOT.1997.614284.

[16] "Paradigma robótico", *HiSoUR Arte Cultura Historia*, 2020. [Online]. Available: https://www.hisour.com/es/robotic-paradigm-43211/. [Accessed: 05- Jul- 2020].

[17] "RoboCup Federation official website", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/. [Accessed: 05- Jul- 2020].

[18] "Robocup", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Robocup. [Accessed: 05- Jul- 2020].

[19] R. Murphy, R. Murphy and R. Arkin, *Introduction to AI robotics*, 8th ed. MIT Press, 2000, 2000.

[20] "RoboCupRescue Robot League", *Rrl.robocup.org*, 2020. [Online]. Available: https://rrl.robocup.org/. [Accessed: 17- Jul- 2020].

[21] "RoboCup Rescue Simulation – RoboCup Rescue Simulation League", *Rescuesim.robocup.org*, 2020. [Online]. Available: https://rescuesim.robocup.org/. [Accessed: 17- Jul- 2020].

[22] A. Simulation, "Agent Simulation – RoboCup Rescue Simulation", *Rescuesim.robocup.org*, 2020. [Online]. Available: https://rescuesim.robocup.org/competitions/agent-simulation-competition/. [Accessed: 17- Jul- 2020].

[23] V. Simulation, "Virtual Robot Simulation – RoboCup Rescue Simulation", *Rescuesim.robocup.org*, 2020. [Online]. Available: https://rescuesim.robocup.org/competitions/virtual-robot-competition/. [Accessed: 17- Jul- 2020].

[24] "Infrastructure – RoboCup Rescue Simulation", *Rescuesim.robocup.org*, 2020. [Online]. Available: https://rescuesim.robocup.org/competitions/infrastructure-competition/. [Accessed: 17- Jul- 2020].

[25] "Simulador de robótica", *HiSoUR Arte Cultura Historia*, 2020. [Online]. Available: https://www.hisour.com/es/robotics-simulator-42971/. [Accessed: 19- Jul- 2020].

[26] "Webots:", *Cyberbotics.com*, 2020. [Online]. Available: https://cyberbotics.com/doc/reference/index. [Accessed: 19- Jul- 2020].

[27] "Webots:", *Cyberbotics.com*, 2020. [Online]. Available: https://www.cyberbotics.com/doc/guide/robots,. [Accessed: 19- Jul- 2020].

[28] "Webots:", *Cyberbotics.com*, 2020. [Online]. Available: https://www.cyberbotics.com/doc/guide/actuators. [Accessed: 17- Jul- 2020].

[29] "Webots:", *Cyberbotics.com*, 2020. [Online]. Available: https://cyberbotics.com/doc/guide/sensors. [Accessed: 17- Jul- 2020].

[30] F. Rosales, "Normativa sobre robots e Inteligencia artificial- notario Fco Rosales", *El Blog de Francisco Rosales | Notario de Alcalá de Guadaíra*, 2020. [Online]. Available: https://www.notariofranciscorosales.com/robots-e-inteligencia-artificial-concepto-y-normativa/. [Accessed: 28- Jul- 2020].

[31] "BOE.es - Agencia Estatal Boletín Oficial del Estado", *Boe.es*, 2020. [Online]. Available: https://boe.es/. [Accessed: 29- Jul- 2020].

[32] "Algoritmo genético", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico#Operadores_gen%C3%A9ticos. [Accessed: 23- Aug- 2020].

[33] "Eolípila", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Eol%C3%ADpila#/media/Archivo:Aeolipile.jpg. [Accessed: 15- Aug- 2020].

[34] "Robot de Leonardo", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Robot_de_Leonardo#/media/Archivo:Leonardo-Robot3.jpg. [Accessed: 15- Aug- 2020].

[35] "Elektro", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Elektro#/media/File:Senator_John_Heinz_History_Center_-_IMG_7802.JPG. [Accessed: 15- Aug- 2020].

[36] *Images.computerhistory.org*, 2020. [Online]. Available: https://images.computerhistory.org/timeline/timeline_ai.robotics_1961.unimate.jpg. [Accessed: 15- Aug- 2020].

[37] "ASIMO", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/ASIMO#/media/File:Honda_ASIMO_(ver._2011)_2011_Tokyo_Motor_Show.jpg. [Accessed: 15- Aug- 2020].

[38] "Nao (robot)", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Nao_(robot)#/media/File:Nao_Robot_(Robocup_2016).jpg. [Accessed: 15- Aug- 2020].

[39] *External-content.duckduckgo.com*, 2020. [Online]. Available: https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fupload.wikimedia.org%2Fwikipedia%2Fcommons%2F0%2F0d%2FLaproscopic_Surgery_Robot.jpg&f=1&nofb=1. [Accessed: 15- Aug- 2020].

[40] *Demandware.edgesuite.net*, 2020. [Online]. Available: http://demandware.edgesuite.net/sits_pod38/dw/image/v2/ABAU_PRD/on/demandware.static/-/Sites-master-catalog-irobot/default/dw74ecbdfe/images/large/R614020.jpg?sw=1000&sh=1000&sm=fit. [Accessed: 15- Aug- 2020].

[41] *External-content.duckduckgo.com*, 2020. [Online]. Available: https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fd.newsweek.com%2Fen%2F

full%2F714523%2Fboston-dynamics-atlas-robot-backflip.png&f=1&nofb=1. [Accessed: 15- Aug- 2020].

[42] *External-content.duckduckgo.com*, 2020. [Online]. Available: https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Fengineering.curiouscatblog.net%2Fwp-content%2Fuploads%2F2012%2F09%2Ftoyota_human_service_robot.jpg&f=1&nofb=1. [Accessed: 15- Aug- 2020].

[43] "RoboCup Humanoid League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/3. [Accessed: 15- Aug- 2020].

[44] "RoboCup Standard Platform League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/5. [Accessed: 15- Aug- 2020].

[45] "RoboCup Middle Size League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/6. [Accessed: 15- Aug- 2020].

[46] "RoboCup Small Size League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/7. [Accessed: 15- Aug- 2020].

[47] "RoboCup Simulation League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/23. [Accessed: 15- Aug- 2020].

[48] "RoboCup Open Platform League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/14. [Accessed: 15- Aug- 2020].

[49] "RoboCup Domestic Standard Platform League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/34. [Accessed: 15- Aug- 2020].

[50] "RoboCup Social Standard Platform League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/15. [Accessed: 15- Aug- 2020].

[51] "RoboCup Logistics League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/17. [Accessed: 15- Aug- 2020].

[52] "RoboCup Soccer League", *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/leagues/18. [Accessed: 15- Aug- 2020].

[53] *Robocup.org*, 2020. [Online]. Available: https://www.robocup.org/system/leagues/images/000/000/002/list/rescue.png?1461148300. [Accessed: 15- Aug- 2020].

[54] *Rescuesim.robocup.org*, 2020. [Online]. Available: https://rescuesim.robocup.org/wp-content/uploads/2017/12/kobe-300x230.png. [Accessed: 15- Aug- 2020].

[55] *Rescuesim.robocup.org*, 2020. [Online]. Available: https://rescuesim.robocup.org/wp-content/uploads/2017/12/rsrss-300x160.jpg. [Accessed: 15- Aug- 2020].

[56] *Rescuesim.robocup.org*, 2020. [Online]. Available: https://rescuesim.robocup.org/wp-content/uploads/2018/01/usar-300x225.jpg. [Accessed: 15- Aug- 2020].

[57] I. Asimov, *Robots E Imperio*. Debolsillo, 2007.

[58] I. Asimov, *Los robots*. Barcelona: Ediciones Martínez Roca, 1984.

[59] R. ASALE, "robótica | Diccionario de la lengua española", *«Diccionario de la lengua española» - Edición del Tricentenario*, 2020. [Online]. Available: https://dle.rae.es/rob%C3%B3tica. [Accessed: 05- Jul- 2020].

[60] "Da Vinci Surgical System", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Da_Vinci_Surgical_System. [Accessed: 16- Jul- 2020].

[61] "Roomba", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Roomba. [Accessed: 17- Jul- 2020].

[62] B. Dynamics, 2020. [Online]. Available: https://www.youtube.com/watch?v=_sBBaNYex3E. [Accessed: 17- Jul- 2020].

[63] "Webots:", *Cyberbotics.com*, 2020. [Online]. Available: https://www.cyberbotics.com/doc/guide/mantis. [Accessed: 19- Jul- 2020].

[64] F. Caparrini and W. Work, "Algoritmos Genéticos - Fernando Sancho Caparrini", *Cs.us.es*, 2020. [Online]. Available: http://www.cs.us.es/~fsancho/?e=65. [Accessed: 03- Sep- 2020].

Anexo 1. English summary

1 Introduction

Most vehicles use wheels as a locomotive system but that's not suitable for every situation or is just not good enough, for this reason alternatives are being studied.

One of the proposed alternatives is the use of zoomorphic robots, which are able to move using legs instead of wheels therefore they're able to adapt better to uneven terrain.

This document aims to cover the use of genetic algorithms to optimize the walking process of an hexapod robot making it possible to explore in extreme scenarios like collapsed buildings or volcanoes.

In order to understand this project more in detail the list of different objectives defined for this work are listed below. Firstly the general objective is:

- **Optimizing the hexapod robot walking process on uneven terrain.** For this to be possible, the motors' behavior has to be defined and optimized in the Webots simulator.

Then, the specific objectives are:

- **Acquiring the necessary skills.** Learning about genetic algorithms and the use of the Webots simulator's functioning in order to be able to face the needs that may appear during the work's development.

- **Implementing the controller.** Make an implementation of the optimal controller which has parameters that can be optimized through genetic algorithms as well as the algorithm itself.

- **Training the genetic algorithm.** Executing training and experimentation necessary to find a solution that satisfies the general objective

- **Evaluating the results.** Rating, through the training data and the final individual, the results obtained and elaborate conclusions about it.

Each and every one of the defined objectives must comply with the Spanish legislation on robotics. This law is the regulation proposed by the European Parliament. In addition, as it is software, it must comply with the intellectual property laws of the system itself.

The methodology followed to carry out this project is an agile methodology, in which the controller and algorithm have been developed over several iterations. Each iteration consisted of a meeting with the tutor in which progress was evaluated and the objectives of the next meeting were proposed and then a work period of about two weeks. This methodology is quite similar to SCRUM but the only differences are that there is only one worker and there is no client but instead the tutor is the one fulfilling this role.

2 State of art

In order to understand the work that must be done in this project, the concept of robotics must be well defined, how it can be classified as, what can it be used for and how you can work with it using simulators. It is also necessary to understand how genetic

algorithms work to be able to comprehend the decisions that have been taken in the different models' development.

2.1 Introduction to robotics

To this day, robotics are meant to do all kinds of tasks in a faster, more efficient and cheaper way, knowing this, the following question arises: What's the definition of robotics?

The term robot comes from the Czech word *robota* and the first appearance of this term in a literary work is attributed to the Czech author Karel Čapek, who used it in his work *"R.U.R. (Robots Universales Rossum)"*.

Eventually, the science fiction and popular science author, Isaac Asimov introduced the word robotics, referring to a field that groups the study of mechatronics, physics and mathematics. This author also enunciated the three well-known laws of robotics.

Otherwise, if a more academic definition is looked for, RAE describes it as: "Technique that applies computer science to the design and use of devices that, in substitution of people, perform operations or work, generally in industrial facilities."

In order to cover the necessary knowledge for this work on robotics, the classification that can be made according to its structure must also be explained.

Knowing this, robots may be classified into five different morphologies:

- **Articulated,** which are static robots that perform actions with tools in a specific working area.
- **Mobiles,** which have great horizontal mobility, their locomotive system is based on wheels and they usually decide their paths through external information they receive or through their own sensors.

- **Androids,** which are the ones based partially or completely on the human body.

- **Zoomorphic,** which are based on animals. This category splits into two new subtypes: based on non-walking animals, and based on walking animals. Last ones are currently being studied to be used as an alternative locomotion method, and also the ones in which this project focuses the most.

- **Hybrids,** which have an architecture that is between one or more of those previously mentioned, making them difficult to fit into one or the other.

2.2 Paradigm

Another crucial point of this process is analyzing the different paradigms that we find in robotics and which define the way robots behave. To define them, the interaction between the three primitives ACT, SENSE and PLAN is commonly used as an explanation for it, each representing the robots actions, the information they acquire from the sensors and the decisions referring to the actions that must be accomplished respectively.

According to this there are three different paradigms:

1. Hierarchical/deliberative paradigm. This paradigm is the oldest and is defined as a SENSE, PLAN and ACT cycle, where each action is defined by a planned course of action upon receiving the information from the sensors. This paradigm uses a closed world model for deciding the actions that must be done.

2. Reactive paradigm. This paradigm arose as a response to the aforementioned, paradigm SENSE-ACT models are made eliminating the PLAN step, in which every action is supported directly by the output of a sensor, allowing more than one action to be carried out because each robot has several models called

behaviors which are chosen based on the input. These results were good in some robots but could not be generalized because of their simplicity.

3. Hybrid paradigm. It's the paradigm we nowadays use. In this paradigm, the idea of a planner is reintroduced but instead of saying how the robot has to act, it divides the robot's objective into several smaller subtasks and assigns behaviors (from the reactive paradigm) and sensors to solve them efficiently. This happens in two steps, first the behaviors are planned and then they are executed.

2.3 RoboCup

The RoboCup is an annual robotics and artificial intelligence competition. It's divided into several categories meant for numerous leagues, the most notable being robot soccer. In addition to soccer, the RoboCup also covers robotics meant for children, robotics within the industry, robotics applied to rescue and robotics within the home.

This competition forecasts that in 2050 a robot soccer team will be capable of winning the future world champions in 2050. This objective is ambitious hoping that the social impact it will have and the progress made will serve to improve society. As it is a very broad objective, in order to accomplish it, more simplistic objectives were created represented by the different competition's categories.

The RoboCup is also proposed as a standard problem to work on, since it can be considered as a further step from the chess' domain and its characteristics are much broader, therefore mastering them is a significant advance on the field.

The official RoboCup categories are:

- **Soccer.** This category is RoboCup's main competition.The researchers' focus consist on cooperative multi-robot and multi-agent systems in dynamically changing conflict environments. In this category robots must be completely

autonomous. Within it there are five leagues, these are separated by the type of robot if it is used.

- Humanoid, In this league different autonomous robots with human form and sensors that simulate human perception are used.
- Standard platform. This league represents the standard problem, since everyone must compete using the NAO robot from SoftBanks Robotics.
- Average size. In this competition, five fully autonomous robots play with a regular-sized ball and teams can create their own robots but the sensors must be on-board and are limited in size and weight.
- Small size. This was the first league created, it's focused on multi-agent systems in dynamic environments with a distributed system problem. Teams can also create their own robots and are limited in size and weight.
- Simulation. This league is focused purely on artificial intelligence and team play, both 2D and 3D simulations are carried out.

- **@Home.** It consists of assisting robots development with the aim of being capable of doing household tasks. This competition is considered a benchmark on the service robots performance in a non-standardized realistic environment.

  It has three leagues, one that covers a standard platform for domestic tasks, another for social environments and also a free platform where you can use your own robots.

- **@Work**. This category includes the competitions that deal with the use of robots for work in industrial environments as well as for logistics applications, where are tested robots that manufacture, refine, assemble or modify something that ends up becoming a finished product.

- **Junior**. It is aimed at encouraging young people who cannot yet participate in adult competitions to become interested in robotics and artificial intelligence, giving them the opportunity to travel to other countries and meet people with common interests. This category is conformed by similar to the rescue and soccer league but simplified and also based on performing on stage assisted by robots.

- **Rescue.** This category focuses on improving both real search and rescue systems in situations that represent catastrophes or natural disasters, as well as simulated systems and even the simulation tools themselves.

  This category has two leagues, one in which real robots are used for competing and the other in which you compete within a simulated environment. The last one splits into 3 sub-leagues, agent simulation, robot simulation and simulators.

2.4 Simulators

A robotics simulator is a program that aims to offer a virtual version of one or more robots making possible worth with it, without the need for the robot itself, lowering costs and allowing it to work at different speeds.

Simulators usually meet a series of characteristics:

- Being able to prototype robots from the simulator itself.
- Having physics engines to be able to generate a realistic movement (usually ODE or Physx
- Having a realistic representation of 3D models, which you can use for standard modeling tools or third-party programs.
- Dynamically simulating robots through scripts in different programming languages.

One of the most used simulators today is Webots which is a free software simulator which has previously been through a paid stage. This simulator is capable of working in C, C++, Python, Java, Matlab and ROS. It has a large amount of documentation and forums, a large number of robots to work on, plus many sensors and actuators.

One of the uses of this simulator is that it allows, using its own library, controlling the entire system through its supervisor class. This permits modifying positions, rotations, speeds or any of the fields of the simulation elements and also restarting the simulation or its controllers, changing its speed or ending it, making it ideal for artificial intelligence applications that require learning.

One of the available robots in Webots is the Mantis. This robot created by Micro Magic is an hexapod which has 3 joints for each leg, these are motors that allows movement between $-\pi$ radians and π radians which represent a 360 degree movement. This is the robot which is used in the project.

2.5 Genetic algorithm

Genetic algorithms are an artificial intelligence technique focused on optimizing systems. It works based on natural evolution models. That is, given a set of individuals in a certain situation, only those that are more adapted to the environment may survive and reproduce, having offspring with shared well-adapted individuals genes who may undergo mutations that could make them better or worse adapting to the environment, ending in the same situation as their parents. In this technique, adaptation to the environment is defined as a function that gives different values to the individuals who are accomplishing best the required task and after a certain number of generations, the algorithm's final result is best adapted individual.

3 Analysis and design

For this project it was used the following software:

- The Operating System used was Windows 10 Home 64 bits.
- The Simulator in which the work was implemented is Webots, which, as previously mentioned, is open software.
- The language in which the controller was implemented is Python.
- The libraries used were the following: controller (simulator), random, math, numpy and deap.

On the other hand, the requirements analysis covers the components initialization, communication between controllers, the stability of the robot, the execution of the genetic algorithm and the storage of the experimentation data as well as the best individual.

4 Implementation of the algorithm

At first, while implementing this project, a model which reacted to the information from the sensors was chosen, this being the motor's position defined by the results sensors activation addition multiplied by a determined weight associated to each one of them. The results were not usable since when touching the ground the sensors were activated but while raising the legs, the value of the sensors changed and then it was unable to make a consistent movement.

To solve these problems, it was decided to use a sine function for each motor to perform a cyclical movement.

4.1 One controller

When the function which defines the behavior of the motors was corrected, a solution based on a single controller was implemented. It started by implementing an associated

phase, amplitude and displacement for each motor as the individual definition thus resulting in 54 genes.

At first the results were normal since although the optimization was not the best, it was consistent but as the experiments progressed and especially from the second half of the second experiment the results began to lose all kinds of consistency.

The previously mentioned error was maintained until changing intro to a two controllers model, but in the fifth experiment the results obtained still had the same error even if the values seemed adequate because they were masked by a recently implemented elitism mechanism.

This error came from a failure in the sensor reading when restarting the simulation, since the sensor references from which the data were obtained stopped giving new values and repeated the last one before the first restart. It happened because the referenced elements had changed within the simulation and weren't updated upon restarting.

4.2 Two controllers

Later, the model was changed to have two controllers. One of them was in charge of the robot and the other one of the simulation supervisor, additionally the number of genes was reduced. That was possible because with the information provided by one motor of each type was sufficient to make the movement consistent, the only change that was necessary to do was shifting the phase so the movement would make sense and also wouldn’t repeat the same on every leg.

This change was meant for being capable of restarting the controller and the simulation without interrupting the evolution process, therefore it was possible to restart the references to the aforementioned sensors, solving the previously mentioned error.

Switching between models meant that the evaluations were slower since it not only had to reset the simulation values but also reset the elements of one of the controllers. With this system, communication between controllers was carried out using receivers and transmitters inside the simulator itself.

During this phase, several approaches were tested, varying throughout the generations both the individual structure as previously mentioned as well as the mutation and crossover probabilities. Therefore finding several possible individuals who could meet the objective but not completely. Some of them moved straight forward but had their legs extended and operated very slowly, others had their legs curled up but were deviated into different directions and finally others were optimized but certain part of their body

None of the individuals in this model fulfilled the general objective to use it as a final individual, and due to this it was decided to make a change in the model so that a better optimization could be carried out and a more adequate result could be reached.

4.3 Final model

In the final model the number of genes were reduced once again from 25 to 15, maintaining only the ones necessary for the displacement and amplitude of each motor type and one for each one of half of the motors phases corresponding to the all of the legs of one side. The same value for the motor of the other side but out of phase by π radians, making them do a complementary movement.

Following this change in the model, in no more than 300 generations it was possible to optimize an individual who was able to walk forward properly without turning or losing stability. This individual was the one the best result in his fitness function so far and the fastest. This result was the first to really fulfill the general objective.

5 Conclusion

The objectives of the work have been achieved, since it's evident that through the practice acquired in its development, the necessary fluency using the simulator has been obtained. It was also demonstrated by solving the different problems that have appeared throughout the process. The controller that was chosen to solve the problem is the result of different iterations that have been improved for a quicker and more efficient optimization. The final solution has been the result of training over hundreds of hours, until an adequate result was obtained. An analysis of the results obtained according to sets of experiments was done, talking about the behavior of their fitness and the behaviour of their best individual.

It is important to mention how notorious became that although the use of genetic algorithms is a very powerful tool for optimizing behaviors, it is also very necessary defining a correct model that allows optimization to be efficient, since in the end it depends on probability and this has a noticeable influence on the results.

On the other hand, although the result is acceptable, it is not optimal. That's because although the robot is capable, in fact, of walking straight on uneven terrain, it would still be necessary that the robot adapted the individual behavior of each leg to the shape of the terrain, which could be the next step in development.

In addition, as previously mentioned, this work was focused on being able to iterate more on this robot and adapting it to rescue tasks, therefore a future work would be to add some type of actuator that allows it to interact with terrein elements.

In conclusion, following this rescue trend mentioned above, it would be necessary to add cameras and some kind of artificial vision system that allows locating and distinguishing elements in the environment where the robot is operating.